PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-097218
(43)Date of publication of application : 08.04.1997

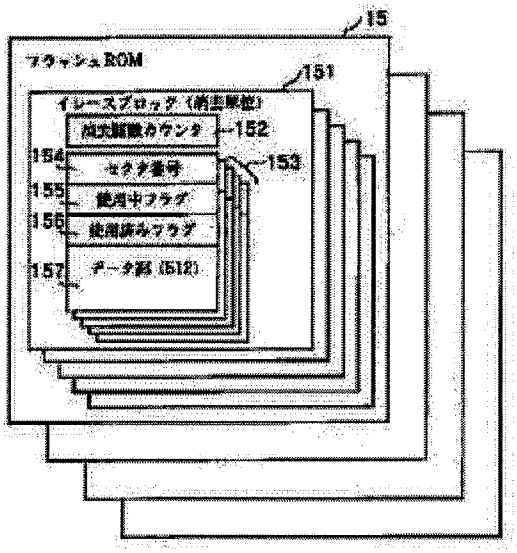
(51)Int.Cl. G06F 12/16
G06F 12/00
G11C 16/06

(21)Application number : 07-251238
(22)Date of filing : 28.09.1995

(71)Applicant : CANON INC
(72)Inventor : OGAWA TAKESHI

(54) METHOD AND DEVICE FOR MANAGING FLASH ROM AND COMPUTER CONTROL EQUIPMENT

(57)Abstract:
PROBLEM TO BE SOLVED: To decrease dispersion in the number of times of erasure for every erasure unit concerning a managing system with which a flash ROM can be made adaptive to a filing system.
SOLUTION: Plural sectors composed of data areas and managing areas corresponding to these data areas are formed in the flash ROM and state information (such as a sector number 154, occupied flag 155 and used flag 156) showing the storage states of respective sectors is stored in the managing areas so that the memory management of the flash ROM can be performed. When the instruction to start garbage collection processing is given for putting invalid data inside the flash ROM in order, while referring to an erasure time counter 152, the erase block of an erasure object is decided. When erasing processing is executed to the decided erase block of the erasure object, the erasure time counter 152 of the erase block is incremented by one.



* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]
[Claim 1]A flash ROM controlling device comprising:
A management tool which forms in a flash ROM two or more storage blocks which comprise a data area and a management domain corresponding to a data area, stores in a management domain state information which shows a memory state of a data area in each storage block, and manages access of a flash ROM based on this state information.
An erasing means which extracts a storage block to which valid data exists in a data area about erase blocks in said flash ROM based on said state information, moves the contents, and eliminates the erase blocks concerned.
A counting means which counts for every erase blocks by making into a number of erase times the number of times to which elimination by said erasing means was performed, and memorizes this number of erase times to a storage area of erase blocks.
A determination means to determine erase blocks which serve as an erasing object in said erasing means based on said number of erase times.

[Claim 2]The flash ROM controlling device according to claim 1, wherein said determination means determines erase blocks with few said numbers of erase times as an erasing object among erase blocks containing invalid data.
[Claim 3]A flash ROM controlling method comprising:
A managing process which forms in a flash ROM two or more storage blocks which comprise a data area and a management domain corresponding to a data area, stores in a management domain state information which shows a memory state of a data area in each storage block, and manages access of a flash ROM based on this state information.
An erasing process which extracts a storage block to which valid data exists in a data area about erase blocks in said flash ROM based on said state information, moves the contents, and eliminates the erase blocks concerned.
A count process of counting for every erase blocks by making into a number of erase times the number of times to which elimination by said erasing process was performed, and memorizing this number of erase times to a storage area of erase blocks.
A decision process which determines erase blocks which serve as an erasing object in said erasing process based on said number of erase times.

[Claim 4]The flash ROM controlling method according to claim 3, wherein said decision process determines erase blocks with few said numbers of erase times as an erasing object among erase blocks containing invalid data.
[Claim 5]Are a computer control system which reads a predetermined program from a memory medium and controls a computer, and said memory medium, Two or more storage blocks which comprise a data area and a management domain corresponding to a data area are formed in a flash

ROM, A procedure code of a managing process which stores in a management domain state information which shows a memory state of a data area in each storage block, and manages access of a flash ROM based on this state information, A procedure code of an erasing process which extracts a storage block to which valid data exists in a data area about erase blocks in said flash ROM based on said state information, moves the contents, and eliminates the erase blocks concerned, A procedure code of a count process of counting for every erase blocks by making into a number of erase times the number of times to which elimination by said erasing process was performed, and memorizing this number of erase times to a storage area of erase blocks, A computer control system provided with a procedure code of a decision process which determines erase blocks which serve as an erasing object in said erasing process based on said number of erase times.

[Translation done.]

* NOTICES *

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Field of the Invention]This invention relates to the controlling method, device, and computer control system of a flash ROM in a computer etc.
[0002]
[Description of the Prior Art]Although a flash ROM has a thing various type now, there are some which were developed for BIOS of the type and personal computer which roughly divided and were developed for flash plate DISK.
[0003]An elimination unit is general 512 bytes in a hard disk, and compatibility of the former with a file system is dramatically good. An elimination unit cannot perform the latter flash ROM only by the big block unit of 64K etc. There is what has the voltage of 12V grade required as write voltage like PROM. Although the latter flash ROM could obtain cheaply, since compatibility with a file system was bad, it was not able to use especially as an archive medium of small capacity.
[0004]
[Problem(s) to be Solved by the Invention]As mentioned above, although the elimination unit is large and compatibility with a file system is bad, the flash ROM designed for BIOS is cheap, and easy to come to hand. Therefore, if such a flash ROM is applicable to a file system, the archive medium of cheap small capacity can be provided.
[0005]This invention provides the flash ROM controlling method and device which make it possible to fit the large flash ROM of an elimination unit with a file system.
[0006]It is possible to overwrite new data to the data area concerned, or to cancel the data of the data area concerned, and to write data in other data areas with a file system, when updating the data memorized by the flash ROM. In a flash ROM, since overwrite of data cannot be performed, data write to other data areas is performed. As a result, invalid data is accumulated and the utilization efficiency of a flash ROM falls. Therefore, when managing access of a file system and the flash ROM which suited, it is indispensable to eliminate the invalid data on a flash ROM.
[0007]As mentioned above, when managing that the large flash ROM of an elimination unit should be applied to a file system, execution of the erasing operation in this elimination unit becomes indispensable. However, a bias produces selection of the elimination unit which is the target of such erasing operation, for example in the elimination unit with much invalid data in an elimination unit to which erasing operation will be carried out if it carries out by being based only on it being few. This bias makes the time life of each elimination unit vary.
[0008]In light of the above-mentioned problems, this invention is a thing.
In the managing system which can adapt the purpose to a file system, it is providing the flash ROM controlling method which can decrease dispersion in the number of erase times for every elimination unit, a device, and a computer control system.

[0009]
[Means for Solving the Problem]A flash ROM controlling device of this invention for attaining the above-mentioned purpose is provided with the following composition. Namely, two or more storage blocks which comprise a data area and a management domain corresponding to a data area are formed in a flash ROM, A management tool which stores in a management domain state information which shows a memory state of a data area in each storage block, and manages access of a flash ROM based on this state information, An erasing means which extracts a storage block to which valid data exists in a data area about erase blocks in said flash ROM based on said state information, moves the contents, and eliminates the erase blocks concerned, It counts for every erase blocks by making into a number of erase times the number of times to which elimination by said erasing means was performed, and has a counting means which memorizes this number of erase times to a storage area of erase blocks, and a determination means to determine erase blocks which serve as an erasing object in said erasing means based on said number of erase times.
[0010]Said determination means determines erase blocks with few said numbers of erase times as an erasing object preferably among erase blocks containing invalid data. It is because decentralization of a number of erase times between erase blocks can be attained and it can act as Mr. [of the rewriting durability / ****] by giving priority to and eliminating erase blocks with few numbers of erase times including invalid data.
[0011]
[Embodiment of the Invention]With reference to attached Drawings, an embodiment of the invention is described below.
[0012][Embodiment 1]
<Composition of camera system> drawing 1 is a block diagram showing the composition of the camera system in Embodiment 1. This camera system comprises an electronic camera and the personal computer 22 connected with the electronic camera at this via the removable external storage 17, the PC communication interface 19, and the PC communication interface 19 so that communication was possible.
[0013]1 is a lens and 2 is a CCD unit which outputs the light which passed along the lens 1 as an electrical signal. 3 is an A/D converter and changes the analog signal from CCD unit 2 into a digital signal. 4 is a speed signal generator unit and supplies a synchronized signal to CCD unit 2 and A/D converter 3. 5 is CPU and realizes various kinds of control in this camera system.
[0014]6 is a signal-processing accelerator and realizes signal processing at high speed. 7 is a cell and 8 is a DC to DC converter for supplying the electric power from the cell 7 to the whole electronic camera. 9 is a power supply controller unit and controls DC to DC converter 8. 10 is a microcomputer which controls panel operation, a display, and a power supply. 11 is a display which displays various kinds of information on a user, and a liquid crystal panel etc. are used. 12 is a control panel and contains the release switch in which a user does a direct control.
[0015]13 is ROM and stores system programs, such as OS. 14 is DRAM and is the main memory of this electronic camera. 15 is a flash ROM and is used as a built-in storage. As for 16, external storage, such as an ATA hard disk, and 18 are extended bus interfaces the interface part of a PCMCIA card, and 17. 19 is a PC communication interface, connects a personal computer etc., and delivers and receives data. 20 is a DMA controller and 21 is a stroboscope. 22 is a personal computer and performs communication with an electronic camera via the PC communication interface 19.
[0016]<Photographing operation> The operation at the time of photography of this electronic camera is explained briefly. If a user pushes the release switch of the control panel 12, CPU5 will detect that and it will start a photographing sequence. It is premised on all the following operations being performed by control by CPU5.
[0017]Now, speed signal generator4 drives CCD2 by the depression of a release switch. The analog signal outputted from CCD2 is changed into a digital signal by A/D converter 3. The DMA transfer

of the output of A/D converter 3 is carried out by DMA controller 20 DRAM14. When the DMA transfer for one frame is completed, CPU5 starts a signal-processing sequence.
[0018]In a signal-processing sequence, a signal-processing program is read from the flash ROM 15 on main memory (DRAM14), the data on main memory is transmitted to the signal-processing accelerator 6, and signal processing is performed. However, the signal-processing accelerator 6 is an arithmetic circuit which helps the processing etc. of the processing which does not perform all of signal processing and is performed by CPU5 which especially time requires, cooperates with the processing software of CPU5 and operates. It records as a graphics file that a part or all of signal processing is completed on the flash ROM 15. Compression will also be performed if the file format recorded at this time makes compression processing **.
[0019]A signal-processing program is one of the files which a file system manages in the flash ROM 15. The program of the camera is dedicated to ROM13 together with OS or the file system. The program of a camera recognizes the file of a specific file name to be a program.
[0020]Since the file is discontinuously arranged in the flash ROM 15 and also the file system of this embodiment rearranges frequently, CPU cannot carry out immediate execution of the ********** in the flash ROM 15. Therefore, it must read to main memory (DRAM14) and it must be performed. The main memory must not be the software supposing being stored in a specific address, in order that a memory manager may do the allocation of the memory location dynamically. Therefore, the file of the program which performs signal processing by this embodiment serves as form like drawing 41.
[0021]Drawing 41 is a figure explaining the stored condition of the control program to the flash ROM in this embodiment. In drawing 41, an identification code is a code for checking that a file is a program. The file is expressed as a set of a variable-length record. There is ID which identifies the kind of information first stored in the record concerned in a record, and the value which shows the size of the record below is stored.
[0022]And the record of a program and the record of relocation information are stored in the file. A program code is data like drawing 42. Drawing 42 is a figure showing an example of the program code expressed with the relative address. In drawing 42, although the 0050th street has jump instruction, CPU recognizes this command to be the jump instruction to an actual address. The operand of this command is expressed by the relative address.
[0023]The data of the relocation information record of drawing 42 is dedicated in form like drawing 43. That is, the address table showing the program address of the data (data which is relative address expression) which must be changed into an actual address in the program of drawing 42 is stored as relocation information.
[0024]If the field for loading the file of drawing 41 to main memory is secured, the memory manager of OS of ROM13 will determine an address. A memory manager's allocation is a function equivalent to an alloc function in the C language. When a memory manager assigns the 8710th street to a program, a program is loaded like drawing 44. Drawing 44 is a figure showing the program code at the time of mapping the program of drawing 41 to the 8710th street of main memory. The operand of jump instruction is transposed to the actual actual address. The program which does the work referred to as reading a program to main memory is stored in ROM13, carrying out conversion to this real address.
[0025]By constituting as mentioned above, signal-processing software and compression software are storable in a storage by a file format. As a result, after a camera reaches final users' origin, the correspondence to various file formats, such as a new signal processing algorithm, BMP form of Windoiws (trademark), TIFF form or form of newly appearing in the future, is attained.
[0026]As mentioned above, the electronic camera in this Embodiment 1 files a taken image in the flash ROM 15.
[0027]<Device driver interface> drawing 2 is a figure showing the layered structure of the file system in the electronic camera of this embodiment. The top layer is the user application 101. The

user application 101 is software which runs by the inside of an electronic camera, and after they open and write a file by a file name, it is closed. [reading and]
[0028]File system API layer 102 is directly called by a function call from the user application 101. This file system API layer 102 has associated and managed the drive name and the file system. Since it constitutes for the business which mounts a file system architecture layer for every drive, it is possible to make two or more file system architecture intermingled.
[0029]It is a portion which performs file management with the actual file system architecture layer 103. The lowest layer is the block device layer 104. The file system architecture layer 103 has realized file input and output using the service which the block device layer 104 provides. In this block device layer 104, data is managed in the unit of a sector and one sector is 512 bytes. The difference in the input/output control for every device and the difference among parameters, such as a head and a cylinder, are absorbed in this block device layer 104. Since it constitutes in this way, the device of two or more kinds can be made intermingled simultaneously.
[0030]Especially in the electronic camera of this embodiment, it has the feature in the Storage and File Management Sub-Division method of the flash ROM in the block device layer 104.
[0031]Although some flash ROMs 11 shown by drawing 1 are type [various] now, there are a type which roughly divided and was developed for flash plate DISK, and a type developed for BIOS of a personal computer. An elimination unit is general 512 bytes in a hard disk, and compatibility of the former with a file system is dramatically good. An elimination unit cannot perform the latter flash ROM only by the big block unit 64k etc. There is what has the voltage of 12V grade required as write voltage like PROM. However, the flash ROM of the latter type is cheap and easy to receive. According to this embodiment, though it is a flash ROM with the feature like the latter, the same service as a hard disk is provided to a file system.
[0032]<Flash ROM driver interface> The service which a block device generally provides to a file system is following two. That is, it is the writing from the sector specified by the read-out (2) logical sector number from the sector specified by (1) logical sector number. And if there is a function of the sector which was specified by (3) logical sector number in addition to this of opening, since the driver of a flash ROM becomes possible [eliminating an unnecessary sector if needed], it can eliminate a flash ROM efficiently.
[0033]Since it can be positively deleted from a cache list if it is a system with cash, although the function mentioned to (3) is an unnecessary function in the usual DISK, it is effective in raising the hit ratio of cash as a result. A file system notifies the sector which became unnecessary to a device driver using the function of (3) by elimination of a file, etc. Although elimination of a flash ROM is the processing which requires time dramatically, in order to hardly consume CPU time, it is good to carry out by background processing.
[0034]Although it explains also in the below-mentioned <FAT cash>, the cash of this embodiment discards the oldest data in a cache list, when accessing new data (data which is not on cash). A possibility that effective data will be discarded from cash by what an unnecessary sector is moved to the last of a cache list for (that is, the data accessed in ancient times moves behind cash) becomes low. A possibility that the intermediary file which should be eliminated remains in cash in the system which generates many especially intermediary files, such as a compiler, is high, and the above-mentioned cache management is dramatically effective in improvement in a hit ratio.
[0035]Drawing 3 is a figure showing the declaration which described the management block of the device driver by the C language. Next of a structure is a ring pointer to the following device, and it is used in order to search the device in a memory. DevName is used as a name of a device. InitDev is a pointer to the initialization routine of a device. ShutDown is a pointer to the shutdown routine of a device. ReadSector is a pointer to the routine which specifies a logical sector and transmits the contents of the medium to a buffer. WriteSector is a pointer to the program (it writes in) which specifies a logical sector and transmits the contents of the buffer to a medium. ReleaseSector is a pointer to the routine which specifies a logical sector and releases a sector.

[0036]A file system will use a device driver mediating this structure. In the case of the fixed disk or the floppy disk, the pointer to the program which does not work at all is substituted for ReleaseSector. Or the pointer which deletes a specified sector from the cache list of a disk cache may be used.
[0037]Data writing to a <flash ROM controlling method> flash ROM is performed in a sector unit from the file system of the upper layer. Drawing 4 is a figure showing the example of the sector structure on a flash ROM. In drawing 4, 151 is an erasure block. This erasure block 151 is a unit of elimination, and it is called a sector in the technical term of a flash ROM. However, in order to distinguish from the "logical sector" which is a unit which a file system treats, it will be called an erasure block here.
[0038]According to drawing 4, two or more flash ROMs 15 are carried in the system, and each flash ROM 15 is constituted by two or more erasure blocks 151. Each erasure block 151 expresses signs that it is constituted by the number-of-erase-times counter 152 and two or more sectors 153. The number-of-erase-times counter 152 is a portion used in order to count the number of times which eliminated the erasure block 151. Each sector 153 has a management domain and a data area. A management domain comprises the flag 155 and the used flag 156 showing the use as a sector having been completed during the sector number 154 showing a logical sector number, and the use showing whether the sector is used effectively or not. The data area is constituted by 512 bytes of data division 157.
[0039]There is no necessity of a data area and a management domain adjoining each other, and arranging, and collecting like drawing 5 and managing is also considered. Drawing 5 is a figure showing the composition which separates and stores the data for management domains, and a data area. Each sector number 154 of two or more sectors 153 is stored in a sector number table. The flag 155 and the used flag 156 are stored in a flag table during use. The contents of the data division 157 are stored in a data table. As for a management domain and the data area corresponding to this at least, although it is also possible to take the above data configurations, dedicating in the same erasure block is preferred.
[0040]A system evaluates "the used flag 156" preferentially from "it is the flag 155 during use." Drawing 6 is a figure showing the meaning corresponding to the state of each flag. FALSE takes the same value as the state after elimination among a figure. If the used flag 156 is FALSE even if the flag 155 is TRUE during use, the data of the sector concerned is invalid.
[0041]Like PROM, in order to rewrite data, once it eliminates a <logical sector rewriting of flash ROM> flash ROM, it must carry out re-writing. And the minimum unit of elimination is large (for example, 64 K bytes), and blanking time is long (for example, 1 second). Then, when the file system of the upper layer tends to rewrite a specific sector, the data rewrite of a logical sector is realized on appearance, without carrying out erasing operation by moving a logical sector to an eliminated field.
[0042]Drawing 7 is a figure explaining the rewriting procedure of a sector. Using the figures, rewriting of the No. 8 sector (a logical sector number is a sector of 8) is made into an example, and is explained in detail. Among drawing 7, left-hand side rewrites, and it is in a front state (state of (a)), and is in the state (state of (b)) after right-hand side rewriting. moreover -- in drawing 7, the number in a management domain expresses a logical sector number -- (under use) -- under use -- as for both the state of FALSE, and [(used)], in the used flag 156, the flag 155 and the used flag 156 show [the flag 155] the state of TRUE during use by TRUE.
[0043]The data of the No. 8 sector is stored in the place of "the sector number 8 (under use)." The No. 8 sector is used as FAT or a part of file, and suppose that the demands rewriting of the No. 8 sector occurred from the upper layer to change the contents now. If demands rewriting occurs, the device driver of a flash ROM will search the unused sector of a flash ROM, will store the data after a sector number and updating for the place as a place of the new No. 8 sector, and will set a flag to TRUE during use. Next, the used flag of the sector which was the No. 8 sector before is set to

TRUE. Rewriting of the data of the No. 8 sector is realized in such a procedure.
[0044]<Garbage collection> When rewriting of the logical sector is performed by the above methods, almost all the fields of a flash ROM will be someday used as "a used sector." Then, it is necessary to once eliminate a flash ROM to a certain timing, and to return "a used sector" to an "unused sector." Operation of a fundamental garbage collection is explained using drawing 8. Drawing 8 is a figure explaining garbage collection operation of the flash ROM in this embodiment.
[0045](A) is in the state before a garbage collection among a figure. In order to explain simply, the flash ROM of this example shall comprise an erasure block of the size for six sectors. A used sector is [the sector of those with three piece and a number of erase times] 5 times during three pieces and use at an erasure block (1) (the contents of the number-of-erase-times counter 152 are 5). One piece and a used sector are one piece, and the sector of those with four piece and a number of erase times is [an unused sector] 9 times during use at an erasure block (2). A garbage collection is started from this state.
[0046]First, an adjustment object erasure block is selected. An adjustment object erasure block turns into an erasure block here as an object which eliminates. When selection of the erasure block for arrangement is preferentially chosen from the erasure block containing many used sectors, its arrangement efficiency is good. However, if the method of setting aside the case where a used sector is not included and making preferentially little erasure block of a number of erase times applicable to arrangement is taken, the erasure block in a chip can be used on the average, and rewriting durability can be distributed. The details of a selection procedure are later mentioned using a flow chart.
[0047]Suppose that the erasure block (1) was selected as a candidate for arrangement now. Next, a sector (in a flag, at TRUE, a used flag is a sector of FALSE during use) is moved to other erasure blocks during use of the erasure block (1) which is arrangement contrast. During use, like the time of rewriting of a sector, the move procedure of a sector searches the unused sector under other erasure blocks, copies the contents of the data area and management domain in a sector during use, and sets the used flag of a sector to TRUE during use of a moved material. Processing in case there is no unused sector is described later.
[0048](B) of drawing 8 is in the state where all sectors were moved to the erasure block (2) during use of an erasure block (1). As a result, only a used sector will not exist in an erasure block (1).
[0049]Next, the erasure block which comprises only a used sector is searched. Here, it searches because all the sectors within an erasure block may be used sectors by chance in the case of the usual rewriting operation. Then, it eliminates to the searched erasure block. Although elimination takes time, since two or more erasure blocks are simultaneously eliminable, it is good to eliminate two or more erasure blocks as at once as possible. An end of elimination will write the thing before elimination carried out value+1 to a number-of-erase-times counter. It is garbage collection completion now.
[0050](C) of drawing 8 is in the state at the time of the end of a garbage collection. Since it is more efficient to eliminate an erasure block as simultaneous as possible, as long as there are a used sector and an unused sector, it is good to adjust much erasure block simultaneously. If it even arranges even if the used sector is not included, if there is what has a value of a number-of-erase-times counter extremely less than other erasure blocks, distribution of rewriting durability can be aimed at. Since the arrangement of data will change once it arranges, it becomes a cause of rewriting durable distribution.
[0051]Although a used sector is in a system [<when there is no unused sector>] next, a garbage collection procedure in case the unused sector has completely been lost is explained using drawing 9. Drawing 9 is a figure explaining operation of a garbage collection in case an unused sector does not exist.
[0052]First, according to the fundamental garbage collection procedure mentioned above, an erasure block (1) is chosen as a candidate for arrangement. Next, the unused sector for moving a sector

during use of an erasure block (1) is searched. When there is an unused sector, a sector is moved like an above-mentioned fundamental garbage collection.
[0053]On the other hand, if there is no unused sector as a result of search, the allocation of the memory block of a size required for evacuation of data will be carried out from the heap area of DRAM14. And a sector is copied to DRAM14 during the use within an adjustment object erasure block. In this case, unlike the case where a sector is moved to another field of a flash ROM, the used flag of the original sector is not set to TRUE. It is because the data in DRAM disappears and restoration of data becomes impossible, when accidents, like the cell 7 of an electronic camera separates at this time happen. (B) of drawing 9 is expressing the sector copied to the field of DRAM14. If all sectors can be evacuated during use of an erasure block of an adjustment object, the erasure block of the adjustment object will be chosen and eliminated (refer to (C) of drawing 9). After elimination finishes, many unused sectors should be made. Therefore, the data which searched a free space next and in which it had shunted to DRAM14 is restored. (D) of drawing 9 shows the state where the garbage collection was completed.
[0054]Even when a garbage collection is carried out in the above-mentioned procedure, data cannot be restored if accidents, like the cell 7 of an electronic camera separates after eliminating an erasure block before restoring data happen. That is, the direction which does not take the method of evacuating the data of a sector to DRAM14, as much as possible can maintain the safety of a system more. An unused sector will be made if a garbage collection is performed once using DRAM. Therefore, if the garbage collection using DRAM14 is performed once and the usual garbage collection is performed after that, although blanking time starts too much, it can improve safety. conversely, the evacuation to DRAM14 -- positive -- carrying out (for example, as long as there is a heap area, it evacuates) -- since the erasure block which can be arranged simultaneously increases, efficiency can be raised. Therefore, it may constitute so that it can specify to which priority shall be given between safety and efficiency. It may constitute so that it may change to efficiency priority automatically, when the power supply of the system is supplied from the cell 7 and it is supplied from safety priority and an AC adapter. This processing is later mentioned with reference to drawing 36.
[0055]If 1 preparation > remaining capacity decreases < erasure block extremely too much, garbage collections will occur frequently and the performance of a system will fall extremely. If an erasure block is used too much only one rather than the erasure block count which can store a part for the total logical sector, it is possible to avoid such a situation. If the case where the one same sector is rewritten 10 times is made into an example when all the sectors become under use noting that 127 sector storing can be carried out per 1 erasure block and there is no excessive erasure block, the writing of ten elimination and 1270 sectors will occur. However, if the erasure block is prepared one too much, only the writing of ten sectors will be generated.
[0056]Therefore, in this embodiment, since the number of erasure blocks is decided by composition of a chip, the total number of logical sectors in which at least one erasure block remains is designed.

[0057]A <timing of garbage collection> garbage collection requires time dramatically, in order to be accompanied by erasing operation. Therefore, usage ****** of a camera will be influenced by when a garbage collection is performed. For example, if a garbage collection is performed when it is not necessary to take a photograph for [, such as a self-timer,] several seconds, a user will not sense stress.
[0058]On the <memory location management on RAM> flash ROM 15, since a sector number and the actual memory location are not related, in order to write a specific sector, the flash ROM 15 must be searched. Then, if the memory location management table showing the stored address of each sector in the flash ROM 15 is created on DRAM114 when a system reboots, reading and writing of high-speed data are realizable to the flash ROM 15. Once it creates a memory location management table, It is possible to always maintain a right memory location management table only

by updating the memory site of a memory location management table only within the case where change arises, to the memory location by the writing and garbage collection of a sector to the flash ROM 15.

[0059]Drawing 10 is a figure explaining the memory location management table created on DRAM. The created memory location management table 140 on RAM was shown in right-hand side among the figure. The value (NULL) as which zero sector and four sectors mean a memory location absence is contained. These are the sectors which there was no writing to the sector after a format, or the file system opened wide.

[0060]Operation of a device driver when a file system takes out with elimination of a file, etc. the command which releases the sector which became unnecessary to a driver is as follows. First, with reference to the pointer of a sector with which the memory location management table 140 on DRAM14 was specified, an applicable sector present in use [on the flash ROM 15] is discovered. And the used flag of the sector concerned is set to TRUE, and an absent value (NULL) is assigned to the pointer of the specified sector of the memory location management table 140 on DRAM14.

[0061]When having shunted the contents of the sector to DRAM14 for a garbage collection, the pointer to DRAM is substituted as the memory location. It is desirable to also dedicate the lock variable for forbidding the concurrent operation to the same logical sector to a table.

[0062]It is convenient when data exchange on a storage can be performed with a <file decompression of MS-DOS> book electronic camera, and the personal computer 22. The flash ROM managing system explained by this embodiment can be used, and MS-DOS (trademark) which has spread with the present personal computer, and a compatible file system can be mounted. The utility which restores the file eliminated once is attached to MS-DOS. However, in this embodiment, in order to raise the erasing efficiency of a flash ROM, it has the composition that the data division of the eliminated sector is lost. It has composition which cannot perform theoretically restoring the medium eliminated with the camera with a personal computer.

[0063]Such an accident can be prevented if the file decompression function in a personal computer can be forbidden. At this embodiment, a file decompression function is forbidden by destroying the data used when MS-DOS is file decompression. This is explained how.

[0064]In MS-DOS (trademark), if a file is eliminated, an empty slot will be made to a directory. Information, including the cluster of a file name / time stamp / beginning, etc., is stored in the directory. Drawing 45 is a figure showing the feature of a directory slot. EndOfDir which shows that it is the last of a list is stored in the last of a directory slot.

[0065]Now, if File B is deleted, the head of a file name will be transposed to the symbol showing deletion, and the cluster chain of FAT will be eliminated. This situation is shown in drawing 46.

[0066]An undeletion program tries restoration of a file based on the information which remained in the 2nd slot. Conversely, restoration of a file can be prevented if this information does not exist.

[0067]Drawing 47 expresses the state after eliminating a file with the DOS compatible file system of this embodiment. It constitutes from this embodiment so that the entry of the file which wants to eliminate the file stored in the last of a directory entry table may be overwritten and EndOfDir may be overwritten at the portion which was a file of the last of a directory entry table. By carrying out like this, restoration of the file by a file decompression function can be prevented.

[0068]At the time of elimination of a file, it leaves the data of the sector as it is like MS-DOS (a sector is not opened), and there is also the method of eliminating a garbage, collecting at the time of a garbage collection and making it FAT with reference to the relation of data.

[0069]In the flash ROM which pretreatment > Exists in the < background, the erasing processing of the direction where the data before elimination is "0" can be carried out at high speed. The check of the completion of elimination of a flash ROM is performed by data polling like the time of data writing. Therefore, when using such a flash ROM, performance can be raised by performing "pretreatment" which rewrites to 0 the data of the sector which became "used" by background processing. If it is made to perform as a task of the lowest priority, it will not lead to the fall of a

throughput.
[0070]The efficiency of pretreatment can be raised if the flag is prepared for "pretreated sector" management in this pretreatment background.
[0071]Therefore, it is efficient when the management flag which can display four "the states where it has pretreated" as a state which a sector can take in addition to "intact", "under use", and ["used"] is prepared inside [of a flash ROM] a sector.
[0072]Drawing 38 is a flow chart showing the control procedure of pretreatment for the improvement in erasing processing speed in this embodiment. In the figure, the sector into which it is used and pretreatment cannot be managed with Step S2501 is extracted. This is realizable by the management flag in a sector serving as "used", and extracting the sector [a sector / "finishing / pretreatment /"]. In Step S2502, overwrite is started for data "0" to the extracted sector. At Step S2503, it is judged whether pretreatment was ended about the sector concerned. Since the writing to a flash ROM is a 1-byte unit, the writing of the number of bytes for one sector is needed. If pretreatment to the sector concerned is not completed, it progresses to Step S2504, and control is moved to other tasks.
[0073]Since this processing is performed by a task with the lowest priority as mentioned above, when CPU5 is in an idle state, this processing is performed again. In this case, processing returns to Step S2503. At this time, if the last writing is not completed, processing is shifted to other tasks as it is.
[0074]After finishing the writing of "0" to all the bytes of the sector concerned as mentioned above, it progresses to Step S2505 from Step S2503, and the management flag of the sector concerned is set to the state which shows "finishing [pretreatment]." And succeedingly, in order to pretreat about other sectors, it returns to Step S2501.
[0075]In a <FAT cash> book system, the memory location is changed into the degree of write-in generating, and an "unused sector" occurs at every time. Then, if there is cash which buffers a portion with much frequency in use preferentially especially, it will be expected that total writing frequency decreases sharply. As there are many memories prepared as cash, they are better, but there is a limit in the memory of a system.
[0076]Although the probability of the data of a sector frequently-used originally which exists in cash is also high, when a sector with low frequency in use is written in large quantities, naturally it will be breathed out from cash.
[0077]Then, if it constitutes so that the cash advance of the management domain which a file system manages may be carried out preferentially, improvement in a throughput is expectable. It is because the management domain of the file system is updated frequently.
[0078]Since one cluster comprises one sector in 720k or the format form of 1.4M in the case of the FAT system of MS-DOS which has spread with the personal computer, even when reading a file sequentially, FAT must be read once to 2 times. When writing a file, still more FAT access occurs. For this reason, if opened by much file in a system, the hit ratio of cash will fall.
[0079]Although based also on application software, the cash which made only FAT management in the FAT system can secure an equivalent hit ratio by one half of memories to the cash for the whole DISK. Drawing 11 is a figure showing hierarchical positioning of cache software. The software of cash serves as a file system and an interim place of a flash ROM like drawing 11.
[0080]Drawing 12 is a figure showing the data structure on the main memory of cash. The whole buffer is managed by uni-directional linear list structure. Data is old at the order of the search direction. If a logical sector number accesses in order of 12, 11, 6, and 5, it will become turn as shown in drawing 12. When the change flag is formed in each sector and it has renewal of data on cash, a change flag changes from FALSE to TRUE. The read-out procedure and the write-in procedure of such FAT cash are explained with reference to drawing 13 and 14. Drawing 13 is a flow chart showing the read-out procedure of FAT cash. Drawing 14 is a flow chart showing the write-in procedure of FAT cash.

[0081]In drawing 13, read-out of N sector is started at Step S1501. It is judged at Step S1502 whether N sector is FAT. If it is not FAT, data will be read from the flash ROM 15 at Step S1509.
[0082]On the other hand, if N sector is FAT at Step S1502, it progresses to Step S1503, and a cache list is searched. Here, the uni-directional linear list explained by drawing 12 will be searched. If N sector exists in cash, it will progress to Step S1507 and data will be read from the buffer of N sector.
[0083]When N sector does not exist in a cache list at Step S1503, it branches to Step S1504 and discharge of the data (drawing 12 data of the sector number 12) of the sector which is not accessed for a long time is performed. First, in Step S1504, the change flag of the paragraph of the last of a cache list is judged. If a change flag is TRUE, it progresses to Step S1505 and contents of change are written in the flash ROM 15. there is no change (when a change flag is FALSE) -- it is -- control is moved to Step S1506 as it is. It is more efficient not to perform writing operation as much as possible until the discharge whose buffer is cash happens although it may regard writing in in a read-out procedure as strange.
[0084]The contents of the N sector are read from the flash ROM 15 to the buffer of the list last at Step S1506. Data is read from the buffer of N sector at Step S1507. The buffer of N sector is moved to the head of a cache list at Step S1508. This is attained in drawing 12 by changing the value of the "following buffer" (address which shows the following buffer) which each sector has. By operation of Step S1508 being repeated whenever access to FAT cash is performed, the buffer which is not accessed automatically shifts from the head of a list toward the last. Therefore, the buffer of the list last is chosen at Step S1504 in order to breathe out the oldest buffer.
[0085]Next, a write-in procedure is explained with reference to drawing 14.
[0086]The writing of N sector is started at Step S1600. In Step S1601, it is judged whether N sector is FAT. If it is not FAT, it will progress to Step S1608 and the writing of the data to the flash ROM 15 will be performed.
[0087]On the other hand, if N sector is FAT at Step S1601, it will progress to Step S1602 and a cache list will be searched. If N sector exists in cash, it will progress to Step S1606 and the writing of data will be performed to the buffer of N sector.
[0088]When N sector does not exist in a cache list at Step S1602, it branches to Step S1603, and N sector is registered into cash while performing discharge of the sector which is not accessed for a long time from a buffer. First, the change flag of the paragraph of the last of a cache list is judged at Step S1603. If a change flag is TRUE, contents of change are written in a flash ROM at Step S1604, and it progresses to Step S1605. If there is no change, control will be moved to Step S1605 as it is (if a change flag is FALSE). Let the paragraph of the last of a cache list be N sector in Step S1605. Then, the writing of data is performed to the buffer of N sector at Step S1606.
[0089]Then, the buffer of N and a sector is moved to the head of a cache list at Step S1607. It is more efficient not to perform writing operation as much as possible until the discharge of cash happens although it may regard not performing the writing to a flash ROM in a write-in procedure as strange.
[0090]Although it is judgment of FAT in Step S1501 and Step S1601, the place of a FAT area can be pinpointed in analyzing the contents of write data also by the system by which an IC card etc. cannot share the information on the upper layer (file system) thoroughly. It is because it is decided to be stored in the portion equivalent to logical sector O in information, including the position of FAT, etc.
[0091]All the (a management domain is included) reading and writing to the flash ROM 15 <write-in [to a flash ROM / 1 byte of]> are eventually performed with 1 byte of reading-and-writing command. The writing of the flash ROM 15 takes the same time as the usual PROM. The writing to the same chip is not made until 1 byte of writing is completed. There are a chip with which the signal wire is prepared as a write end signal, and a chip with which the special signal is not prepared. In the case of the latter, a write end must be checked by the technique called data polling. Data

polling is a method very just like verification, and is the busy control method for which it waits until write data and read data are in agreement.
[0092]When a write end can be known with a signal wire, it can use together with interruption to CPU5, and can write in, and the central-processing-unit time in waiting can be assigned to another task.
[0093]As mentioned above, data polling must be performed when it is a chip without a signal wire. In order to gather the efficiency of data writing, it must write in in pipeline to a lot of chips, and the loss of data polling time must be pressed down. Therefore, before 1 byte of writing is completed, it is necessary to put control into the next operation. Before performing new reading and writing, it is good to check whether former writing is completed. Drawing 15 expresses the situation by the C language.
[0094]The 1st line of drawing 15 is an entrance of the function which performs data writing. The pointer to a structure for the first argument to save the address and data which were written in at the end, the address which the 2nd argument writes in, and the 2nd argument are data to write in.
[0095]In the 3rd line, the data written to the chip with reference to the address written in at the end is compared with the data written to the last, and a loop is executed until both are in agreement. This is data polling. Completion of the last writing will slip out of this loop.
[0096]Data is written in a new address by the 4th line. The address and data which were written to be the 5th line by the 6th line this time are saved. This information is used by the next data polling.
[0097]RotateRdyQueue of the list of the 7th line is a system call of an operating system which yields CPU to the task of the ready condition of the same priority that should be performed by the next of a self-task.
[0098]The 9th line is an entrance of a read-out function. The pointer to a structure for the 1st argument to save an address and data and the 2nd argument are addresses to read. This function returns the data stored in the address specified by the 2nd argument to the program of a higher rank.
[0099]In the 11th line, since the value returned if the address which it tried to read is an address written in at the end is the data written in at the end, it returns the information saved in the structure. The 12th line is the same data polling as the 3rd line. Unless it succeeds in data polling, another address of the same chip cannot be read. The contents of the address which data polling finished and was specified by the 13th line are returned.
[0100]If the writing to one chip is carried out to the above composition, it can see certainly only by writing in with a chip number and increasing a task, and the upper drawing speed can be raised. It is effective, when it is made not to process until a buffer is covered with the contents which prepare and (if it is two chips two sectors) write in the sector buffer for a chip number purposely, in order to raise the whole throughput.
[0101]The characteristic place of the program of drawing 15 is carrying out before the next writing rather than performing data polling immediately after data writing. Therefore, the address written last time and the RAM area where data is saved are secured for every chip, and it stores as a structure "struct DEV."
[0102]Drawing 39 is a flow chart showing the write-in procedure of 1 byte data to the flash ROM in this embodiment. This flow chart shows the control procedure of the writing to one flash ROM chip. In Step S2601, it is judged whether the last write-in processing was completed. If the last write-in processing is not completed, it progresses to Step S2604, and control is moved to other tasks as it is.
[0103]On the other hand, if the last write-in processing is completed, the following write data is prepared and this is saved to DRAM14. A judgment of the write-in end in the above-mentioned step S2601 is made by comparison with the data written in the flash ROM, and the data held at this step S2602.
[0104]Then, the writing of data is started in Step S2603. According to the above processings, when

writing in by two or more tasks to two or more flash ROM chips, the writing processing which applied what is called a round-robin system becomes possible, and data can be efficiently written in to two or more ROM tips. The control program of multitasking is stored in above-mentioned ROM13. It incorporates, and as real-time OS [like], VxWorks (trademark), pSOS (trademark), etc. are marketed and such real-time OS [like] is stored in ROM13.

[0105]There are a chip which needs the special write voltage of 12V grade like PROM in the case of the writing of <sharing of flash ROM write-in power supply> data or elimination, and a chip with which writing becomes high-speed by giving write voltage. It will be connected with the cost hike of an electronic camera, if voltage generation sections, such as a DC to DC converter for exclusive use, are provided when using such a chip. However, there is a portion which needs special voltage, such as charge of a stroboscope, a working part, and a drive of CCD, in a camera conventionally, and the DC to DC converter etc. are carried. Then, by performing the write voltage of a flash ROM, strobe charging, and a mechanism drive by Time Division Multiplexing, a system can be built with the DC to DC converter of small capacity, and the cost of a system can be pressed down.

[0106]Drawing 16 expresses the program which manages a power supply so that the output capacitance of a DC to DC converter may not be exceeded by the C language. Line 1-6 is a zoom-in function of one step, and Line 7-13 is a 1 sector writing **** write-in function in a flash ROM. A zoom-in function gains the semaphore for the resource control of a DC to DC converter "SemDCDC" by Line3, and calls 1 step ** or ****** for a motor by Line4. After a motor drive finishes, the semaphore for the resource control of a DC to DC converter "SemDCDC" is opened. A semaphore is a general method for managing resources with the operating system of multitasking, and many operating systems are preparing it as a system call.

[0107]That is, supposing "SemDCDC" was already used by other tasks by Line3, execution of the task which was going to zoom in until other tasks opened the semaphore "SemDCDC" will be suspended.

[0108]A write-in function gains a semaphore "SemDCDC" by Line9, and writes the data of one sector in a flash ROM. If it checks that data polling was performed by Line11 and the last writing has been completed, a semaphore "SemDCDC" will be opened by Line12. Thus, if a program is constituted, performing the writing of zoom-in and a flash ROM simultaneously will be lost. Zoom is 1 step unit, and since writing is a sector unit, it can certainly acquire a power supply after very short holding time.

[0109]When drawing 16 is explained further, Line1 of drawing 16 is an entrance of the function which zooms in. There is no argument in this function. One right of SemDCDC declared as a use right of a power supply by Line3 is acquired. At this time, if one does not have a use right, execution of the task which called this function will be suspended. If another task releases the use right of a power supply, the task which called ZoomUp will return to ready condition again. And the function to which the motor of Line4 is moved can be called. And by Line5, a power supply use right is returned and work of this function is ended. Line7 is an entrance of the function which writes the data of one sector in EEPROM, acquired the Line9de power supply use right, and has returned it by Line12.

[0110]Drawing 40 is a flow chart for explaining the share procedure of the power supply mentioned above. In the figure, it is judged whether supply of the output power of DC to DC converter 8 by the switch controller 9 was released at Step S1701. In Step S1702, the contents of directions for power supply reservation are analyzed, and it branches to Step S1703, and 1705, 1707 or 1709 according to this directions result.

[0111]If the contents of directions are supply of CCD drive electric power, they will progress to Step S1703 and will supply the electric power for a CCD drive to CCD2. And at Step S1704, if the end (namely, end of photographing operation) of a CCD drive is detected, it will progress to Step S1711 and a power supply will be released. If it is a charging request of a stroboscope, it will progress to Step S1705 and the charging power to the stroboscope 21 will be made to provide to the switch controller 9. And if charge of a stroboscope is completed at Step S1706, it will progress

to Step S1711 and a power supply will be released. Whenever the electric power supply of charge charges predetermined time, it releases a power supply for other current supply. That is, the program which manages the charge to the stroboscope 21 exists in a predetermined task separately, and completion of charge is managed by the task.

[0112]If the contents of directions are the drives of a zoom mechanism, they will progress to Step S1707 and will perform an electric power supply to the drive system (un-illustrating) of a zoom mechanism. And at Step S1708, if the zooming operation of one step is finished, it will progress to Step S1711, and a power supply is released. If the contents of directions are the writing to a flash ROM, it will progress to Step S1709 and the write-in electric power to the flash ROM 15 will be supplied. If the writing for one sector finishes, it will progress to Step S1711 from Step S1710, and a power supply will be released.

[0113]In Step S1704, and 1706, 1708 and 1710, although it waits for the end of each operation, in this waiting loop, the control to other tasks moves and a multitask operation is carried out. Since this management processing can be started at any time and may be simultaneously started by two or more tasks from each task, it is checking power supply release at Step S1701.

[0114]According to the flow chart of the above drawing 40, it becomes possible to use a power supply by time sharing. However, it is one program which depended and had all the systems (CCD / stroboscope / zoom / flash ROM). If such software is developed, the cost of development / debugging / maintenance will become large, and it will become difficult to maintain extendibility and pliability.

[0115]Then, development efficiency can be raised by using the resource control function which likens a power supply with one resources and OS provides. Then, resource control by an above-mentioned semaphore is performed. That is, each control program of the actuator of CCD, the actuator of a stroboscope, the actuator of zoom, and the actuator of a flash ROM can assign the power supply by which time sharing was carried out by gaining and releasing resources (semaphore) called a power supply.

[0116]Drawing 48 is a figure explaining time sharing use of the power supply by this embodiment. If it is in the state where the power supply semaphore was released when a power supply demand occurs by a certain task A (for example, CCD) as shown in the figure, the semaphore is gained and a power supply is occupied (Steps S2001-S2003). Then, in Step S2004, if the electric power supply from the power supply concerned is obtained and predetermined processing is performed, it will progress to Step S2005 and a semaphore will be released.

[0117]At the task B which required power supply acquisition later than the task A on the other hand, by the power supply demand in Step S2011, a semaphore cannot be gained but it becomes the release waiting of a semaphore by Step S2012. And if a semaphore is released from the task A, the task B gains this semaphore and a power supply is occupied (Step S2013). Predetermined processing is performed by the task B after that (Step S2014), and a power supply is released (Step S2015).

[0118]Time sharing use of a power supply is attained by management of the power supply resources by the above semaphores.

[0119]According to drawing 48, there is only one semaphore which shows the use right of power supply resources, but it cannot be overemphasized that it may be made for two or more semaphores to exist.

[0120]<Operation explanation of electronic camera of embodiment> drawing 17 is a flow chart showing the operation procedures to the start of service from reboot of this embodiment. If a system reboots at Step S101, the management domain of the flash ROM 15 will be scanned at Step S102, and the memory location management table 140 will be created on DRAM14. In parallel to this processing, it is counted and set to sector counters during the intact sector counters on DRAM14, used sector counters, and use how many there is any sector corresponding to each state. This counter is updated when it is behind operated to the flash ROM 15, and it is used for judging

memory efficiency. Then, it progresses to Step S103 and various kinds of services are started.
[0121]Drawing 18 is a flow chart showing the procedure of read-out service of a specified sector. First, read-out of N sector is started at Step S201. N sector is locked in Step S202. The lock of a sector is performed using a lock variable. This lock variable is managed with the memory location of each sector with the memory location management table 140. In Step S202, when the sector is already locked by other tasks, after waiting and other tasks unlock being unlocked by other tasks, the sector concerned is locked. A self-task can occupy the locked sector until it unlocks it at Step S206.
[0122]If a logical sector is locked at Step S202, with reference to a memory location management table, it will be checked [effective in the sector concerned] at Step S203 whether data storage is carried out. When effective data is recorded and there is nothing, it branches to Step S204. In Step S204, dummy data (for example, all 0 etc.) is read as contents of the sector. When it is judged that effective data is stored at Step S203, it branches to Step S205. At Step S205, data is read from a flash ROM (or main memory) based on the value of a recording place management table.
[0123]Here, when a garbage collection is under execution and N sector is evacuated to main memory (DRAM14), the pointer of the memory location management table makes main memory pointing. The portion enclosed with a figure middle point line is a period which occupies N sector. Since the safety of one sector operation is guaranteed according to such a locking mechanism, it is possible to read freely the sector which is not under operation in the middle of a garbage collection.
[0124]Drawing 19 is a flow chart showing the procedure of write-in service of a logical sector. The writing of N sector is started at Step S301. A logical sector is locked like Step S202 at Step S302.
[0125]Next, it is judged whether a memory location management table is searched with Step S303, and data effective in N sector is recorded at it. If effective data is recorded and it will not be recorded on Step S304, it will branch to Step S305, respectively. In Step S304, the data of the flash ROM (or main memory) currently recorded as effective data till then is canceled. Processing of the data cancellation in Step S304 adds explanation in detail using the flow chart of drawing 21. Control moves to the back step S305 of Step S304.
[0126]In Step S305, the storage area for writing in N sector in the flash ROM 15 is gained. The acquisition procedure of the storage area in Step S305 adds explanation in detail using drawing 23. If it succeeds in acquisition of a storage area normally at Step S305, control will be moved to Step S308. At Step S308, the data of N sector is written in the field of the gained flash ROM 15.
[0127]On the other hand, at Step S305, when there is no memory location in a flash ROM (i.e., when acquisition of a storage area goes wrong), it branches to Step S306. Step S306 gains main memory to evacuation of data. The memory management function which an operating system provides performs partitioning of main memory. This is a function which is equivalent to an alloc function by the C language. And the secured field is managed according to uni-directional linear list structure.
[0128]Drawing 20 is the appearance of the evacuation data list gained on main memory. (a) is in the state which does not have data in an evacuation data list, and END_OF_LIST is substituted for the list. (b) is in the state where the contents of each sector of the sector numbers 3 and 20,221 are evacuated to the evacuation data list.
[0129]A recording place management table is updated at Step S309. Here, the pointer to the recorded flash ROM (or main memory) is substituted. A logical sector is unlocked at Step S310. the period surrounded by the figure middle point line -- the logical sector can be occupied. Step S311 estimates memory efficiency. About the evaluation procedure of memory efficiency, explanation is added in detail using the flow chart of drawing 21. When memory efficiency gets worse as a result of evaluation of memory efficiency, control is moved to Step S312. The garbage collection mentioned above is performed at Step S312. About a garbage collection, explanation is added in detail with reference to the flow chart of drawing 24. The writing of N sector is completed at Step S313, and it returns to the main routines.
[0130]The pointer (address on bus space) of the memory location is dedicated by the memory

location management table. From the next field (in the figure, shown immediately downward) of "the pointer to the following data" of the data which shunted to the main memory of (b) of drawing 20, there are the data structure and compatibility on the flash ROM in the left-hand side of drawing 10. The pointer to this compatible portion is dedicated by the memory location management table. By constituting in this way, it becomes possible to treat a flash ROM and main memory with a single algorithm by the read-out program side of data.

[0131]Next, the procedure (above-mentioned step S304) of canceling memory of the specified sector is explained. Drawing 21 is a flow chart showing the procedure of canceling memory.

[0132]Memory cancellation of the appointed field is started at Step S401. In Step S402, it is judged whether the field which memorizes the specified sector is on main memory. If it is on main memory, it will branch to Step S405. The appointed field is deleted from a shunting sector list (uni-directional linear list shown by drawing 20 in this example) at Step S405.

[0133]The deletion procedure of the appointed field from a uni-directional linear list follows a list in order of the search direction from the head of a list first, and detects the paragraph to which the pointer is carrying out pointing of itself. And it realizes because he assigns now the value made into pointing to the pointer of this detected paragraph. And the main storage area deleted from the list is returned to an operating system at Step S406. Return of the storage area to an operating system is a function equivalent to the free function of the C language.

[0134]on the other hand, the field specified at Step S402 does not come out on main memory -- it is (namely, on a flash ROM) -- it branches to Step S403. In Step S403, the management flag of the sector on the specified flash ROM is changed into "used." This is attained by setting a used flag to TRUE. At Step S404, one value of the intact sector counters on main memory is decreased. It returns at Step S407.

[0135]Next, the evaluation procedure (Step S311) of memory efficiency is explained. Drawing 22 is a flow chart showing the evaluation procedure of memory efficiency.

[0136]Evaluation of memory efficiency is started at Step S501. Step S502 compares the value of intact sector counters and the value of used sector counters which were set as main memory. Here, whether the value of used sector counters is the same to the value of intact sector counters, and when it exceeds, it constitutes so that aggravation of memory efficiency may be reported to a higher rank program (Step S502, S504). If the value of intact sector counters is larger than the value of used sector counters, an evaluation result will be normalized and will be returned normally (Step S503).

[0137]Next, the acquisition procedure (Step S305) of the storage area of a flash ROM is explained. Drawing 23 is a flow chart showing the acquisition procedure of the storage area of a flash ROM.

[0138]Acquisition of the storage area of a flash ROM is started at Step S601. The search right of an unused sector is acquired at Step S602. Here, the search right of an unused sector is managed using the function of the semaphore which an operating system provides. Here, only the processing term surrounded by the dotted line from Step S602 to the step S609-/step S611 can monopolize the search right of an unused sector. It is the structure for preventing the situation where two or more tasks gain the same field simultaneously.

[0139]A pointer is moved to the sector of the beginning of a flash ROM at Step S603. With reference to the management flag (under use a flag, a used flag) of the sector, the condition of use of the sector concerned is judged at Step S603. If it is [be / it] under use in used, it branches to Step S605. If the sector pointed at at Step S605 now is the last sector, it branches to Step S611. In this case, since an usable field will not exist in the flash ROM 15, after opening the search right of an unused sector wide at Step S611, an abnormal return is carried out at Step S612. If the sector pointed at at Step S605 now is not the last sector, it will branch to Step S606. In Step S606, since a pointer is moved to the following sector, it returns to Step S604.

[0140]If the management flag of the sector which step S604 pointer shows is intact, it will branch to Step S607. In Step S607, the management flag of a flash ROM is changed "into use" (a flag is set to

TRUE during use). And one value of the intact sector counters provided in main memory is decreased at Step S608. In this case, since it has succeeded in acquisition of the storage area to a flash ROM, the search right of an unused sector is wide opened at Step S609, and it returns normally at Step S610.

[0141]Next, the procedure of a garbage collection (Step S312) is explained. Drawing 24 is a flow chart showing the procedure of a garbage collection.

[0142]A garbage collection is started at Step S701. In Step S702, the erasure block for arrangement (henceforth, arrangement object block) is elected. About the election procedure of an arrangement object block, explanation is added in detail using the flow chart of drawing 25. In Step S703, the unused sector of an arrangement object block is made used. About the procedure of this used-izing, explanation is added in detail using the flow chart of drawing 26. The purpose of making the unused sector in an arrangement object block making it used first here, Even if it is during a garbage collection, it has composition in which the reading and writing to the sector in which other tasks contain the sector in an arrangement object block are possible, and is to prevent new data from being written in the sector in an arrangement object block by other tasks during a garbage collection.

[0143]In Step S704, a sector is moved to other storage areas (namely, other erasure blocks) during the use in an arrangement object block. About the processing which moves a sector to other storage areas during use, explanation is added in detail with reference to the flow chart of drawing 27.

[0144]In continuing Step S705, elimination of the arrangement object block which ended movement of the sector during use is performed. About the procedure which eliminates an arrangement object block, explanation is added in detail with reference to the flow chart of drawing 28. In elimination of this arrangement object block, the contents of the number-of-erase-times counter 152 are copied to main memory. After finishing elimination of the arrangement object block in Step S705, the data evacuated to main memory at Step S706 is returned to the number-of-erase-times counter of the erasure block of a flash ROM concerned. And it returns from a garbage collection at Step S707.

[0145]Next, the election procedure (Step S702) of the arrangement object block in a garbage collection is explained. Drawing 25 is a flow chart showing the procedure which carries out arrangement object block election.

[0146]First, election of an arrangement object block is started at Step S801. The first erasure block is set to an evaluation pointer at Step S802. Similarly, the candidate pointer for arrangement is set to the first erasure block at Step S803.

[0147]Next, it is judged whether the used sector is contained in the erasure block which an evaluation pointer shows at Step S804. If the used sector is not contained, Step S804 and Step S805 are skipped, and control is moved to Step S807.

[0148]On the other hand, when the used sector is contained in the erasure block which an evaluation pointer shows at Step S804, control is moved to Step S805. Step S805 compares the value of the number-of-erase-times counter of the erasure block which the value of the number-of-erase-times counter of the erasure block which the candidate pointer for arrangement shows, and an evaluation pointer show. If there are few numbers of erase times of the erasure block which an evaluation pointer shows, control will be moved to Step S806. An evaluation pointer is substituted for Step S806 to the candidate pointer for arrangement. If there are more numbers of erase times of erasure Plock which is carried out also at Step S805 and an evaluation pointer shows on the other hand, control will be moved to Step S807 as it is.

[0149]It is judged whether the evaluation pointer shows the last erasure block at Step S807. If it is not the last erasure block, after moving an evaluation pointer to the following erasure block at Step S808, it will return to Step S804. As mentioned above, the candidate pointer for arrangement comes to show little erasure block of a number of erase times by repeating processing of Steps S804-S808 including a used sector.

[0150]When the evaluation pointer shows the last erasure block at Step S807, it branches to Step S809. At Step S809, it returns to garbage collection processing (processing of drawing 24). The erasure block which the candidate pointer for arrangement at this time shows is elected as a candidate for arrangement.
[0151]Next, the processing (Step S703) which makes used the unused sector in the selected arrangement object block is explained. Drawing 26 is a flow chart showing the procedure which makes the unused sector of an arrangement object block used.
[0152]Processing is started at Step S901. At Step S902, a pointer is moved to the sector of the beginning of an arrangement object block. Next, the search right of an unused sector is acquired at Step S903. This has the same effect as Step S602 of the flow chart of drawing 23, and while being surrounded by the dotted line to Step S908, it monopolizes the search right of an unused sector. That is, other tasks are forbidden from carrying out search which is an unused sector until it scans for all the sectors of an arrangement object block and changes an unused sector into a used sector. However, since an unused sector is changed into a used sector by operation of only a management flag, the monopoly time of a search right is short and the whole throughput does not fall.
[0153]At Step S904, it is judged whether the sector which the present pointer shows is an unused sector. If it is an unused sector, it branches to Step S905. The memory is discarded at Step S905. The procedure of Step S905 is as the flow chart of drawing 21 having explained. An unused sector is changed into a used sector by this processing. In Step S906, it is judged whether the pointer shows the sector of the last of an arrangement object block. If the last sector is shown, if that is not right, to Step S908, it will branch to Step S907. At Step S907, a pointer is moved to the following sector and control is returned to Step S904.
[0154]If a pointer is a sector of the arrangement object block last at Step S906, the search right of an unused sector is wide opened at Step S908, and it returns to garbage collection processing (flow chart of drawing 24) at Step S909.
[0155]Next, the processing (Step S704) which moves a sector to the unused sector of other erasure blocks during use of an arrangement object block is explained. Drawing 27 is a flow chart with which the move procedure of a sector is expressed during use of an arrangement object block.
[0156]Processing is started at Step S1000. A pointer is moved to the sector of the beginning of an arrangement object block at Step S1001. The sector to which a pointer points is processed in following Steps S1002-S1012.
[0157]The management flag (under use a flag, a used flag) of the sector concerned is judged at Step S1002. If the value of the management flag has become "under use" at Step S1002, control will be moved to Step S1003, and if it is "used", control will be moved to Step S1012. A logical sector is locked at Step S1003. The locked sector is occupied by a self-task until it is unlocked at Step S1011.
[0158]A storage area is gained at Step S1004. The procedure of reservation of the storage area in Step S1004 is as the flow chart of drawing 23 having explained. Here, each sector in an arrangement object block is processing of the above-mentioned step S703, and since all are made used, the storage area secured serves as an erasure block of those other than an arrangement object block.
[0159]If it succeeds in acquisition of a storage area, processing will progress to Step S1008. In Step S1008, the data of the sector concerned is copied to the gained field. And according to movement of a sector, the memory location management table 140 is updated at Step S1009.
[0160]On the other hand, when acquisition of a storage area goes wrong at Step S1004, it branches to Step S1005. In Step S1005, the storage area for data saving is gained from main memory (DRAM). Acquisition of the storage area for data saving is as Step S306 of the flow chart of drawing 19 having explained. In Step S1006, the data of the sector concerned is copied to the gained field. And the Storage and File Management Sub-Division table is updated at Step S1007. The original memory is discarded at Step S1010. That is, the used flag of the sector to which a pointer points is set to TRUE. And the logical sector concerned is unlocked at Step S1011.

[0161]At Step S1012, it is judged whether the sector to which a pointer points is a sector of the last of an arrangement object block. If it is the last sector, if it is not the last sector, to Step S1014, it will branch to Step S1013, respectively. In Step S1013, a pointer is moved to the following sector, it returns to Step S1002, and above-mentioned processing is repeated about the following sector. In Step S1014, since processing is finished about all the sectors in an arrangement object block, it returns to garbage collection processing (flow chart of drawing 24).
[0162]Next, the erasing processing (Step S705) of an arrangement object block is explained. Drawing 28 is a flow chart showing the elimination procedure of the erasure block used as the candidate for arrangement.
[0163]Processing is started at Step S1101. The number-of-erase-times counter of an arrangement object block is copied to main memory at Step S1102. Elimination of an arrangement object block is performed at Step S1103. In Step S1104, the value to which the value of the number-of-erase-times counter copied to main memory was made to increase one time is written in a flash ROM. That is, the value of the elimination counter of the arrangement object block concerned is made to increase from the value before erasing processing one time. Then, it returns to garbage collection processing (flow chart of drawing 24) at Step S1105.
[0164]The garbage collection processing shown by above-mentioned drawing 24 is the processing which used the flash ROM as much as possible, and its safety of evacuation data is high. However, as the paragraph [above-mentioned / <in case there is no unused sector>] explained, when the data of a sector is positively shunted during use using main memory (DRAM14) and two or more erasure blocks are eliminated, the efficiency of erasing processing is good. However, since data is shunted to DRAM14, safety falls about the data under shunting (for example, when a cell separates and current supply stops, the data which shunted to DRAM will be lost). Then, the classification of a power supply is judged, when a supplied power source is a cell, the safety of shunting data is thought as important, and in the case of an AC adapter, since there are few dangers that current supply will stop, it may constitute so that the efficiency of erasing processing may be thought as important. The processing in this case is explained with reference to drawing 36.
[0165]Drawing 36 is a flow chart explaining the procedure in the case of switching garbage collection processing based on power supply classification. In the figure, about the step which performs the same processing as the processing shown with the flow chart of drawing 24, the same step number is attached and detailed explanation is omitted here.
[0166]If garbage collection processing is started in Step S1300, it will progress to Step S1301 and the gestalt of the current supply to the device concerned will be judged. Here, the switch controller 9 of drawing 1 judges whether the supply origin of a power supply is the cell 7, or it is AC adapter 23, and notifies to CPU5. When power supply classification is the cell 7, it progresses to Step S1304 and garbage collection processing shown by above-mentioned drawing 24 is performed.
[0167]On the other hand, when power supply classification is an AC adapter in Step S1301, it progresses to Step S1302. In Step S1302, processing equivalent to Step S702 of drawing 24, S703, and S704 is performed, and a sector is shunted during used-izing of the unused sector in the elected arrangement object block, and use. And in Step S1303, it judges whether there is any sufficient free space to shunt a sector to main memory (DRAM14), and if there is sufficient free space, it will return to Step S1302. In Step S1302, an arrangement object block other than the last arrangement object block is elected, and above-mentioned processing is repeated.
[0168]If free space sufficient on DRAM14 is lost, it will progress to Step S1304 from Step S1303, and the arrangement object block elected by above-mentioned processing will be eliminated. And the data which shunted to main memory at Step S706 is returned to the flash ROM 15, and this processing is ended.
[0169]As mentioned above, according to processing of drawing 36, when a power supply is supplied by an AC adapter, data is shunted using the availability of main memory positively, two or more arrangement object blocks can be elected and put in block, erasing processing can be performed,

and the efficiency of erasing processing improves.
[0170]Although the gestalt of a garbage collection is automatically switched in the above-mentioned processing based on power supply classification, it cannot be overemphasized that it can switch by a manual by operation of the control panel 12.
[0171]Next, the release procedure of the logical sector which is one of the basic services is explained. Drawing 29 is a flow chart showing the release procedure of a logical sector.
[0172]Release of N sector is started at Step S1201. N sector is locked at Step S1202. As a result, a logical sector can be occupied by a self-task until it is unlocked at Step S1205. Then, memory of the sector concerned is discarded at Step S1203. It is as the flow chart of drawing 21 having explained the discarding treatment of this memory. An "out" value is assigned to Step S1204 to the memory location management table 140 of DRAM14. In Step S1205, a logical sector is unlocked and it returns at Step S1206.
[0173]For example, in general file systems, such as MS-DOS (trademark), when eliminating a file, in FAT, it is only making overwrite possible about the sector belonging to the file concerned, and releasing each sector is not performed. Therefore, when the flash ROM managerial system of this embodiment is applied to such a file system, the data which became invalid will be left behind as an effective sector, and efficiency, such as a garbage collection, is made to fall on a file system. Therefore, the sector which became unnecessary based on directions (for example, file erasure) of a file system is detected, and if it constitutes so that this may be released, the efficiency of a garbage collection can be raised more.
[0174]Drawing 37 is a flow chart showing the release procedure of an unnecessary sector when file erasure is directed from a file system. In the figure, it is judged whether there were any directions of file erasure from a file system at Step S1401. When there are directions of file erasure, the sector contained in the file directed that it should progress to Step S1402 and should eliminate is extracted. Extraction of a sector can be extracted by referring to FAT. And release processing of the sector explained with the flow chart of above-mentioned drawing 29 is performed at Step S1403 about each sector extracted at previous Step S1402.
[0175][Embodiment 2] Embodiment 2 is described below.
[0176]<Disk controller emulation> The feature which looked at the Storage and File Management Sub-Division system of the flash ROM explained by above-mentioned Embodiment 1 from the upper layer resembles the disc medium well. By therefore, the thing to include in the system provided with the emulation function of a disk controller. It becomes possible to transpose a disk controller and a disc medium to the Storage and File Management Sub-Division system (or IC card incorporating the Storage and File Management Sub-Division system of this embodiment) of a disk controller emulation and this embodiment. Although the IC card represented by PCMCIA has spread in recent years, it becomes possible by building the Storage and File Management Sub-Division system of a disk controller emulation function and the above-mentioned Embodiment 1 into an IC card to use as a removal storage. A 2nd embodiment explains what built the Storage and File Management Sub-Division system of Embodiment 1 into the IC card.
[0177]Drawing 30 is a block diagram showing the composition of the IC card in Embodiment 2. In the figure, 200 shows the whole IC card. 201 is a microcomputer and performs a disk controller emulation and Storage and File Management Sub-Division. 202 is ROM and stores the program of the microcomputer 201. 203 is RAM and functions as main memory of the microcomputer 201. 204 is a flash ROM and stores data by the Storage and File Management Sub-Division system explained by the above-mentioned Embodiment 1. That is, the flash ROM 204 is managed in the management domain and data area which were explained by drawing 4.
[0178]205 is a command/data latch part, and holds a command, a cylinder number, etc. from an external bus which were received from the host device. 206 is a FIFO memory, and outputs and inputs data with FIFO. 207 is tuple ROM, it has memorized the feature of the card concerned, etc. and read-out of it is possible only from an external bus.

[0179]The function of each above-mentioned composition becomes clearer by subsequent explanation of operation.
[0180]Drawing 31 is an easy block diagram of the host system for using the IC card of this Embodiment 2. In the figure, 301 is a microcomputer by the side of a host system. 302 is a card interface and connects the internal bus of a host system, and the external bus of IC card 200. The card interface 302 is provided also with the power supply line for performing current supply to IC card 200, and the signal wire for receiving the interrupt request (IRQ output) from IC card 200.
[0181]Drawing 32 is a flow chart which shows the procedure at the time of the host system of drawing 31 connecting an IC card. If processing is started at Step S4100, the current supply to an IC card will be started at Step S4101. In Step S4102, the data stored in tuple form is analyzed from tuple ROM7 in IC card 200. The feature of the IC card connected can be understood in analyzing the contents of tuple ROM7.
[0182]In Step S4103, it is judged whether the IC card connected can connect with an internal bus using the tuple information analyzed at Step S4102. And to Step S4104, if connection is possible, if connection is impossible, it branches to Step S4105, respectively. In Step S4104, the bus by the side of an IC card is mapped to the memory space and IO space of a built-in bus of a host. It will be in the same state as a disk controller is in the space of the bus of a host device at this time.
[0183]Drawing 33 is a flow chart which shows the main sequence of the microcomputer 1 in IC card 200. If the power supply of an IC card is switched on at Step S4201, the Storage and File Management Sub-Division system will be initialized at Step S4202. That is, the state of the logical sector of all the erasure blocks of the flash ROM 204 is once read, and a memory location management table is created to RAM203 for main memory according to the read information. The buffer of ring shape is prepared and initialized as a command buffer on main memory at Step S4203, and interruption processing is permitted. Operation of a interruption routine starts after this processing.
[0184]The sequence of a interruption routine is shown in the flow chart of drawing 34. Since it becomes that it is [explanation of the flow chart of drawing 33] easier to understand operation of a interruption routine, the flow chart of drawing 34 is explained here.
[0185]If a host system writes a command in the address of the command to a command / data latch 205, an interrupt will occur from a command / data latch 205 to the microcomputer 201. The command / data latch 205 is mapped by IO address space of a host bus and the bus inside an IC card, and IO address is assigned as a command/data is shown in drawing 35, respectively. Drawing 35 is a figure showing IO allotment in the command/data latch of this embodiment. In this example, an interrupt occurs to the microcomputer 201 by writing a command (for example, ReadSector which directs read-out of data (s)) in the address of Command in drawing 35.
[0186]If an interrupt occurs, the software of the microcomputer 201 will move control to Step S4301 of the flow chart of drawing 34. In Step S4302, the data written in the command / data latch 205 is read, and data is stored in the ring buffer on main memory. A interruption routine is ended at Step S4303, and it returns to the flow chart of drawing 33.
[0187]It returns to explanation of the flow chart of drawing 33. The microcomputer 201 judges the state of a command buffer at Step S4204. If data is stored in the command buffer, it branches to Step S4205, and if data is not stored, it will branch to Step S4213. CPU is made into hibernation at Step S4213. Although it has the function for many one-chip microcomputers to stop execution of a command, and to cut down the consumed electric current, CPU of this embodiment is also provided with this kind of function. And if the interrupt request signal by IRQ is inputted, CPU201 will return from hibernation and will perform an above-mentioned interruption routine. When execution of an interruption program ends, it returns from Step S4213 and returns to Step S4204.
[0188]If data is stored in the command buffer at Step S4204, it will shift to Step S4205. In Step S4206, data is read from a ring buffer. A command is interpreted at Step S4206. When it is the Seek command, in the case of Step S4207 and the ReadSector (s) command, branch to Step S4208, and,

in WriteSector (s), it branches to Step S4209, and, in the case of the IdentifyDrv command, branches to Step S4210, respectively. Although there are other commands, what is not an explanation overlay important point of this embodiment excluded, and has simplified the flow chart. If execution of the command to step S4207-4210 is ended, it will return to Step S4204, and the above-mentioned processing is repeated.

[0189]The Seek command is executed in Step S4207. Since Seek does not have a head in a flash ROM unlike a disk device, the validity etc. with which the following command is equipped are only checked. When the head position exceeding the number of heads which an IC card supports, etc. are specified, an error occurs like a disk unit.

[0190]Step S4208 performs processing to the ReadSector (s) command. The number of the sector which should read the ReadSector (s) command is specified by SectorCount of drawing 35. therefore, the sector of the place specified in Step S4208 -- a SectroCount piece -- the act to read is performed. Since it is managing in the Storage and File Management Sub-Division system of this embodiment using the linear logical sector number, A linear logical sector number is calculated for a cylinder / head / sector number to origin, the contents of the logical sector are transmitted to FIFO memory 206, and increment of SectorNumber of a command / data latch 205 is also performed. FIFO memory 206 is a FIFO memory used as the composition which reads the data which could read the data written in from the internal bus of IC card 200 from the external bus, and was written in from the external bus from an IC card internal bus.

[0191]Here, an above-mentioned linear logical sector number is explained. The number generally specified to a hard disk is a three-dimensional discontinuous number decided with the parameter of a sector, a cylinder, and a head. For example, when a number of cylinders is a hard disk whose 1024 pieces and number of heads are 16 pieces and whose sector number is 63 pieces, a sector number will be 1024x16x63=1032192 piece.

[0192]It is designed specify all of the three above-mentioned parameters, and access them although I hope that this sector can be accessed as No. 1032192 from No. 0. For example, the next of cylinder 500, head 16, and the sector 63 is condition of accessing cylinder 501, head 0, and the sector 1. Each initial is taken and these three parameters are called a CHS parameter.

[0193]In an operating system like MS-DOS (trademark), although a linear (continuous) sector number is used inside, a device driver changes this into a CHS parameter. In the system of this embodiment, since a linear sector number is used, origin is asked for a linear sector number for the value of a CHS parameter. It is a cylinder number x(16x63)+ head number x(63)+ sector number in the case of the above mentioned hard disk.

A linear logical sector number can be found by calculating.

[0194]Step S4209 performs processing which writes data in the sector of the place specified by the data latch. Data is received from a host system by FIFO memory 206 course.

[0195]Step S4210 performs processing which returns the information whether IC card 200 is carrying out the emulation of what kind of hard disk. That is, processing which writes the data containing the spec. as hard disks, such as a number of cylinders and ModelNumber, in FIFO memory 206 is performed.

[0196]By including in an IC card, the Storage and File Management Sub-Division system explained by Embodiment 1 as beyond <the analysis of the file system> explained can be used for replacement uses, such as an ATA hard disk. However, there is no technique of getting the information for performing processing called opening of the unnecessary sector produced by FAT cash and file erasure, such as the ATA command, from a host system. By carrying out additional mounting of the sector number designation command which carries out cash to the open command of a sector using the empty portion of the ATA command, the function of FAT cash and unnecessary sector release is realizable. And it cannot be overemphasized that it is better to have been able to realize opening of FAT cash or a sector also by systems, such as present MS-DOS which does not assume that there is such a function.

[0197]The FAT system stores the information of the place and size of FAT in the portion equivalent to the logical sector number 0. According to this embodiment, the place and size of FAT are acquired by reading this sector, and it uses for processing of FAT cash. Although it is an IC card which originally should not understand the contents of write data in a similar manner, it can use for processing of an IC card judging and opening an unnecessary sector independently in analyzing the information (a directory entry and FAT) for a file system. It is not the talk restricted to FAT, of course, and if the contents of the data written in also with the file system of HPFS or Macintosh (trademark) are analyzed, detection of an unnecessary sector is possible. It makes it possible to perform optimization processing doubled with operation of the file system also by the interface of an ATA hard disk with constituting in this appearance.
[0198]The storage which made the program of the above-mentioned embodiment memorize can also attain the purpose of this invention attained with the function of the above-mentioned device, or the function of a method. For example, by equipping a personal computer with the storage and executing a flash ROM control program which is explained to the following read from the storage, While being able to use a flash ROM on a par with a disc system, decentralization of rewriting durability can be attained. The structural feature of the program concerning this invention for this is as being shown in drawing 49.
[0199]Drawing 49 is a figure showing the control procedure of the control program stored in the storage in this embodiment, and the memory map of this storage.
[0200]In drawing 49 (a), 350 is management processing and two or more sectors which comprise a data area and a management domain corresponding to a data area are formed in a flash ROM, The state information (under use the sector number 154, the flag 155, the used flag 156) which shows the memory state of a data area in each sector is stored in a management domain, and access of the flash ROM 15 is managed based on this state information. For example, as the flow chart of drawing 18 or drawing 19 showed, the writing and read-out of data to the flash ROM in a sector unit are controlled.
[0201]351 is erasing processing, it extracts the sector to which valid data exists in a data area about the erasure block in a flash ROM based on state information, moves the contents, and eliminates the erasure block concerned. This is the garbage collection processing shown with the flow chart of drawing 24.
[0202]352 is count processing, is counted for every erasure block by making into a number of erase times the number of times to which elimination by the erasing processing 351 was performed, and memorizes this number of erase times at the number-of-erase-times counter 152 of each erasure block. This is processing shown by Step S1102 of drawing 28, and S1104.
[0203]353 is decision processing and determines the erasure block which should serve as an erasing object in the erasing processing 351 based on the value of a number-of-erase-times counter. This is processing shown with the flow chart of drawing 25.
[0204]In a actual control procedure, if the start of garbage collection processing is directed by the management processing 350, the decision processing 353 will determine the erasure block of an erasing object with reference to the number-of-erase-times counter 152. If erasing processing is performed by the erasing processing 351 to the erasure block of the erasing object determined by decision processing, the count processing 352 will ********** the one number-of-erase-times counter 152 of the erasure block concerned.
[0205]The control program for realizing the above-mentioned control procedure is stored in storages, such as a floppy disk, a hard disk, or CD-ROM, with composition as shown, for example in the memory map of (b) of drawing 49. The above-mentioned control program is read, for example by information processors, such as a personal computer, is loaded on main memory (RAM), and is executed by CPU. Loading of the above-mentioned control program to a main memory top may be performed via LAN.
[0206]in addition -- drawing 49 -- (-- b --) -- setting -- management processing -- a module --

350 -- ' -- decision processing -- a module -- 353 -- ' -- erasing processing -- a module -- 351 -- ' -- a count -- a treatment module -- 352 -- '. It is a program module which performs each processing of the management processing 350, the decision processing 353, the erasing processing 351, and the count processing 352 shown with the control procedure, respectively.

[0207]Even if it applies this invention to the system which comprises two or more apparatus, it may be applied to the device which consists of one apparatus. It cannot be overemphasized that this invention can be applied also when attained by supplying a program to a system or a device. In this case, the storage which stored the program concerning this invention will constitute this invention. And the system or device operates by the method defined beforehand by reading the program from this storage to a system or a device.

[0208]

[Effect of the Invention]As explained above, according to this invention, in the managing system which can adapt a flash ROM to a file system, it becomes possible to decrease dispersion in the number of erase times for every elimination unit.

[0209]

[Translation done.]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平9－97218

(43)公開日　平成９年(1997)４月８日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	12/16	３１０	7623－5B	Ｇ０６Ｆ	12/16	３１０Ａ
	12/00	５０１			12/00	５０１Ｈ
Ｇ１１Ｃ	16/06			Ｇ１１Ｃ	17/00	３０９Ｆ

審査請求　未請求　請求項の数５　ＯＬ　（全 42 頁）

(21)出願番号　特願平7－251238

(22)出願日　平成７年(1995)９月28日

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目30番２号

(72)発明者　小川　武志
東京都大田区下丸子３丁目30番２号　キヤノン株式会社内

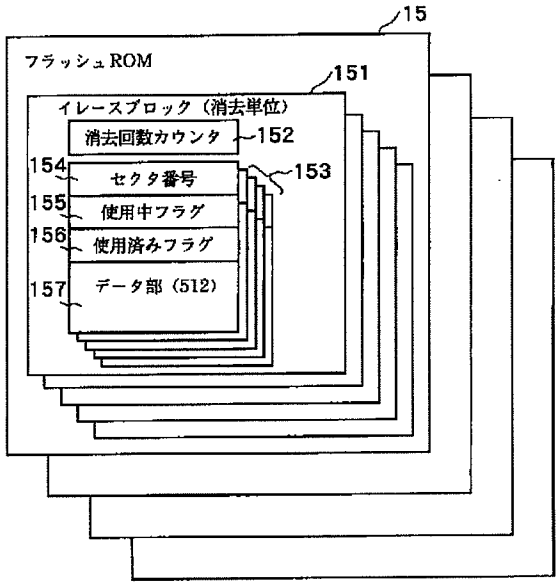
(74)代理人　弁理士　大塚　康徳　（外１名）

(54)【発明の名称】　フラッシュＲＯＭ管理方法及び装置及びコンピュータ制御装置

(57)【要約】

【課題】フラッシュＲＯＭをファイルシステムに適応させることが可能な管理方式において、各消去単位毎の消去回数のばらつきを減少させる。

【解決手段】データ領域とこれに対応する管理領域とで構成される複数のセクタをフラッシュＲＯＭに形成し、各セクタの記憶状態を示す状態情報（セクタ番号１５４、使用中フラグ１５５、使用済フラグ１５６）を管理領域に格納してフラッシュＲＯＭの記憶管理を行う。フラッシュＲＯＭ内の無効データを整理するためにガベージコレクション処理の開始が指示されると、消去回数カウンタ１５２を参照して消去対象のイレースブロックを決定する。決定された消去対象のイレースブロックに対して消去処理を実行すると、当該イレースブロックの消去回数カウンタ１５２が１つインクリメントされる。



1

【特許請求の範囲】

【請求項１】　データ領域、及びデータ領域に対応する管理領域とで構成される複数の記憶ブロックをフラッシュＲＯＭに形成し、各記憶ブロックにおいてデータ領域の記憶状態を示す状態情報を管理領域に格納し、該状態情報に基づいてフラッシュＲＯＭのアクセスを管理する管理手段と、

前記フラッシュＲＯＭ内の消去ブロックについて、前記状態情報に基づいてデータ領域に有効データが存在する記憶ブロックを抽出し、その内容を移動させて当該消去ブロックを消去する消去手段と、

前記消去手段による消去が行われた回数を消去回数として消去ブロック毎にカウントし、該消去回数を消去ブロックの記憶領域に記憶するカウント手段と、

前記消去手段において消去対象となる消去ブロックを前記消去回数に基づいて決定する決定手段とを備えることを特徴とするフラッシュＲＯＭ管理装置。

【請求項２】　前記決定手段は、無効データを含む消去ブロックのうち前記消去回数の少ない消去ブロックを消去対象に決定することを特徴とする請求項１に記載のフラッシュＲＯＭ管理装置。

【請求項３】　データ領域、及びデータ領域に対応する管理領域とで構成される複数の記憶ブロックをフラッシュＲＯＭに形成し、各記憶ブロックにおいてデータ領域の記憶状態を示す状態情報を管理領域に格納し、該状態情報に基づいてフラッシュＲＯＭのアクセスを管理する管理工程と、

前記フラッシュＲＯＭ内の消去ブロックについて、前記状態情報に基づいてデータ領域に有効データが存在する記憶ブロックを抽出し、その内容を移動させて当該消去ブロックを消去する消去工程と、

前記消去工程による消去が行われた回数を消去回数として消去ブロック毎にカウントし、該消去回数を消去ブロックの記憶領域に記憶するカウント工程と、

前記消去工程において消去対象となる消去ブロックを前記消去回数に基づいて決定する決定工程とを備えることを特徴とするフラッシュＲＯＭ管理方法。

【請求項４】　前記決定工程は、無効データを含む消去ブロックのうち前記消去回数の少ない消去ブロックを消去対象に決定することを特徴とする請求項３に記載のフラッシュＲＯＭ管理方法。

【請求項５】　メモリ媒体から所定のプログラムを読み込んでコンピュータを制御するコンピュータ制御装置であって、前記メモリ媒体は、

データ領域、及びデータ領域に対応する管理領域とで構成される複数の記憶ブロックをフラッシュＲＯＭに形成し、各記憶ブロックにおいてデータ領域の記憶状態を示す状態情報を管理領域に格納し、該状態情報に基づいてフラッシュＲＯＭのアクセスを管理する管理工程の手順コードと、

2

前記フラッシュＲＯＭ内の消去ブロックについて、前記状態情報に基づいてデータ領域に有効データが存在する記憶ブロックを抽出し、その内容を移動させて当該消去ブロックを消去する消去工程の手順コードと、

前記消去工程による消去が行われた回数を消去回数として消去ブロック毎にカウントし、該消去回数を消去ブロックの記憶領域に記憶するカウント工程の手順コードと、

前記消去工程において消去対象となる消去ブロックを前記消去回数に基づいて決定する決定工程の手順コードとを備えることを特徴とするコンピュータ制御装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、コンピュータ等におけるフラッシュＲＯＭの管理方法及び装置及びコンピュータ制御装置に関する。

【０００２】

【従来の技術】フラッシュＲＯＭは現在いろいろなタイプのものがあるが大きく分けてフラッシュＤＩＳＫ用に開発されたタイプとパーソナルコンピュータのＢＩＯＳ用に開発されたものがある。

【０００３】前者は消去単位がハードディスクで一般的な５１２バイトであり、ファイルシステムとの整合性が非常に良い。後者のフラッシュＲＯＭは消去単位が例えば６４Ｋなどといった大きなブロック単位でしか行えない様になっている。また、ＰＲＯＭの様に書き込み電圧として１２Ｖ等の電圧が必要なものもある。後者のフラッシュＲＯＭの方が安価に入手できるがファイルシステムとの整合性が悪い為に特に小容量の記録メディアとしては使う事が出来なかった。

【０００４】

【発明が解決しようとする課題】以上のように、ＢＩＯＳ用に設計されたフラッシュＲＯＭは、その消去単位が大きく、ファイルシステムとの整合性が悪いが、安価で入手しやすい。従って、そのようなフラッシュＲＯＭをファイルシステムに適用できれば、安価な小容量の記録メディアを提供することができる。

【０００５】本発明は、消去単位の大きいフラッシュＲＯＭをファイルシステムと適合させることを可能とするフラッシュＲＯＭ管理方法及び装置を提供するものである。

【０００６】ファイルシステムにより、フラッシュＲＯＭに記憶されているデータの更新を行う場合、当該データ領域へ新たなデータを上書きするか、当該データ領域のデータを無効化して他のデータ領域へデータを書き込むことが考えられる。フラッシュＲＯＭでは、データの上書きができないので、他のデータ領域へのデータ書込みを行う。この結果、無効データが蓄積され、フラッシュＲＯＭの利用効率が低下する。よって、ファイルシステムと適合したフラッシュＲＯＭのアクセスを管理する

場合、フラッシュＲＯＭ上の無効データを消去することが不可欠である。

【０００７】以上のように、消去単位の大きいフラッシュＲＯＭをファイルシステムに適用すべく管理する場合、該消去単位における消去動作の実行が不可欠となる。しかしながら、このような消去動作の対象となる消去単位の選択を、例えば消去単位中の無効データの多い少ないのみに基づいて行うと、消去動作が行われる消去単位に偏りが生じる。この偏りは、各消去単位の時間的な寿命をばらつかせることになる。

【０００８】本発明は上記の問題に鑑みてなされたものであり、フラッシュＲＯＭをファイルシステムに適応させることが可能な管理方式において、各消去単位毎の消去回数のばらつきを減少させることが可能なフラッシュＲＯＭ管理方法及び装置及びコンピュータ制御装置を提供することを目的とする。

【０００９】

【課題を解決するための手段】上記の目的を達成するための本発明のフラッシュＲＯＭ管理装置は以下の構成を備える。即ち、データ領域、及びデータ領域に対応する管理領域とで構成される複数の記憶ブロックをフラッシュＲＯＭに形成し、各記憶ブロックにおいてデータ領域の記憶状態を示す状態情報を管理領域に格納し、該状態情報に基づいてフラッシュＲＯＭのアクセスを管理する管理手段と、前記フラッシュＲＯＭ内の消去ブロックについて、前記状態情報に基づいてデータ領域に有効データが存在する記憶ブロックを抽出し、その内容を移動させて当該消去ブロックを消去する消去手段と、前記消去手段による消去が行われた回数を消去回数として消去ブロック毎にカウントし、該消去回数を消去ブロックの記憶領域に記憶するカウント手段と、前記消去手段において消去対象となる消去ブロックを前記消去回数に基づいて決定する決定手段とを備える。

【００１０】また、好ましくは、前記決定手段は、無効データを含む消去ブロックのうち前記消去回数の少ない消去ブロックを消去対象に決定する。無効データを含み、消去回数の少ない消去ブロックを優先して消去することにより、消去ブロック間の消去回数の分散化が図れ、書き換え耐久をぶんさんさせることができるからである。

【００１１】

【発明の実施の形態】以下に添付の図面を参照して本発明の実施の形態を説明する。

【００１２】［実施形態１］

＜カメラシステムの構成＞図１は実施形態１におけるカメラシステムの構成を表すブロック図である。本カメラシステムは、電子カメラと、これに着脱可能な外部記憶媒体１７、ＰＣ通信インターフェース１９、及びＰＣ通信インターフェース１９を介して電子カメラと通信可能に接続されたパーソナルコンピュータ２２から構成される。

【００１３】１はレンズであり、２はレンズ１を通った光を電気信号として出力するＣＣＤユニットである。３はＡ／Ｄコンバータであり、ＣＣＤユニット２からのアナログ信号をデジタル信号へ変換する。４はＳＳＧユニットであり、ＣＣＤユニット２とＡ／Ｄコンバータ３に同期信号を供給する。５はＣＰＵであり、本カメラシステムにおける各種の制御を実現する。

【００１４】６は信号処理アクセラレータであり、信号処理を高速に実現する。７は電池であり、８は、電池７よりの電力を電子カメラ全体へ供給するためのＤＣ／ＤＣコンバータである。９は電源コントローラユニットであり、ＤＣ／ＤＣコンバータ８をコントロールする。１０はパネル操作・表示装置・電源のコントロールを行うマイクロコンピュータである。１１はユーザへ各種の情報を表示する表示装置であり、液晶パネル等が用いられる。１２はコントロールパネルであり、ユーザが直接操作するレリーズスイッチを含む。

【００１５】１３はＲＯＭであり、ＯＳ等のシステムプログラムを格納する。１４はＤＲＡＭであり、本電子カメラの主記憶である。１５はフラッシュＲＯＭであり、内蔵記憶媒体として使用する。１６はＰＣＭＣＩＡカードのインタフェース部、１７はＡＴＡハードディスクなどの外部記憶媒体、１８は拡張バスインタフェースである。１９はＰＣ通信インタフェースであり、パーソナルコンピュータ等を接続してデータの授受を行う。２０はＤＭＡコントローラ、２１はストロボである。また、２２はパーソナルコンピュータであり、ＰＣ通信インターフェース１９を介して、電子カメラとの通信を行う。

【００１６】＜撮影動作＞この電子カメラの撮影時の動作を簡単に説明する。コントロールパネル１２のレリーズスイッチをユーザが押すと、ＣＰＵ５がそのことを検出して撮影シーケンスを開始する。以下の動作は全てＣＰＵ５によるコントロールで行われることを前提とする。

【００１７】さて、レリーズスイッチの押下により、ＳＳＧ４がＣＣＤ２を駆動する。ＣＣＤ２から出力されるアナログ信号は、Ａ／Ｄコンバータ３でデジタル信号へ変換される。Ａ／Ｄコンバータ３の出力は、ＤＭＡコントローラ２０によってＤＲＡＭ１４へＤＭＡ転送される。１フレーム分のＤＭＡ転送が終了した時点でＣＰＵ５は、信号処理シーケンスを開始する。

【００１８】信号処理シーケンスでは、フラッシュＲＯＭ１５から信号処理プログラムを主記憶（ＤＲＡＭ１４）上に読み出し、主記憶上のデータを信号処理アクセラレータ６へ転送し信号処理を行う。但し、信号処理アクセラレータ６は信号処理の全てを行うわけではなく、ＣＰＵ５で行う処理の特に時間のかかる処理などを助ける演算回路であり、ＣＰＵ５の処理ソフトウェアと連携して動作する。信号処理の一部または全部が終了すると

画像ファイルとしてフラッシュＲＯＭ１５へ記録する。この時記録するファイルフォーマットが圧縮処理を必用とするのであれば圧縮も行う。

【００１９】信号処理プログラムは、フラッシュＲＯＭ１５の中でファイルシステムが管理するファイルの１つである。カメラのプログラムはＯＳやファイルシステムといっしょにＲＯＭ１３に納められている。カメラのプログラムは、特定のファイル名のファイルをプログラムであると認識する。

【００２０】フラッシュＲＯＭ１５の中でファイルは不連続に配置されている上に、本実施形態のファイルシステムが頻繁に再配置を行うため、、フラッシュＲＯＭ１５内の制御プログラムををＣＰＵが直接実行することはできない。従って、主記憶（ＤＲＡＭ１４）に読み出して実行させなければならない。更に、主記憶はメモリマネージャが動的に記憶場所をアロケーションするため、、特定のアドレスに格納されることを想定したソフトウエアであってはならない。そのため、本実施形態で信号処理を行うプログラムのファイルは図４１のような形式となっている。

【００２１】図４１は本実施形態におけるフラッシュＲＯＭへの制御プログラムの格納状態を説明する図である。図４１において、識別コードはファイルがプログラムであることを確認するためのコードである。ファイルは可変長のレコードの集合として表現されている。レコードには、始めに当該レコードに格納されている情報の種類を識別するＩＤがあり、次にそのレコードの大きさを示す値が格納されている。

【００２２】そして、プログラムのレコードとリロケーション情報のレコードがファイルに格納されている。プログラムコードは例えば図４２のようなデータである。図４２は相対アドレスで表現されたプログラムコードの一例を表す図である。図４２では、００５０番地にジャンプ命令があるが、ＣＰＵはこの命令を絶対番地へのジャンプ命令と認識する。この命令のオペランドは相対アドレスで表現されている。

【００２３】図４２のリロケーション情報レコードのデータは図４３のような形式で納められている。即ち、図４２のプログラムの中で、絶対番地へ変換しなければならないデータ（相対アドレス表現になっているデータ）のプログラム番地を示すアドレステーブルがリロケーション情報として格納される。

【００２４】図４１のファイルを主記憶にロードするための領域を確保すると、ＲＯＭ１３のＯＳのメモリマネージャがアドレスを決定する。メモリマネージャのアロケーションはＣ言語ではalloc関数に相当する機能である。メモリマネージャがプログラム用に８７１０番地を割り当てた場合図４４のようにプログラムがロードされる。図４４は図４１のプログラムを主記憶の８７１０番地へマッピングした場合のプログラムコードを示す図である。ジャンプ命令のオペランドが実際の絶対番地に置き換えられている。この実アドレスへの変換をしながらプログラムを主記憶へ読み出すと言う作業を行うプログラムはＲＯＭ１３に格納されている。

【００２５】以上のように構成することにより、記憶媒体に信号処理ソフトウエアや圧縮ソフトウエアをファイル形式で格納することができる。その結果、カメラが最終ユーザの元へ届いてから、新しい信号処理アルゴリズムや、Ｗｉｎｄｏｉｗｓ（商標）のBMP形式やTIFF形式、あるいは将来新たに登場する形式等、多種多様なファイル形式への対応が可能となる。

【００２６】以上の様に、本実施形態１における電子カメラは、撮影画像をフラッシュＲＯＭ１５へファイルするものである。

【００２７】＜デバイスドライバインタフェース＞図２は、本実施形態の電子カメラにおけるファイルシステムの階層構造を表す図である。最上位の層がユーザアプリケーション１０１である。ユーザアプリケーション１０１は電子カメラの内部で動くソフトウェアであり、ファイルをファイル名でオープンして読み書きした後クローズする。

【００２８】ユーザアプリケーション１０１から直接ファンクションコールによって呼び出されるのがファイルシステムＡＰＩ層１０２である。このファイルシステムＡＰＩ層１０２がドライブ名とファイルシステムを関連付けて管理している。各ドライブ毎にファイルシステムアーキテクチャ層をマウントする用に構成しているため、複数のファイルシステムアーキテクチャを混在させる事が可能となっている。

【００２９】ファイルシステムアーキテクチャ層１０３が実際のファイル管理を行う部分である。最下位の層がブロックデバイス層１０４である。ファイルシステムアーキテクチャ層１０３がブロックデバイス層１０４の提供するサービスを利用してファイル入出力を実現している。このブロックデバイス層１０４では、データをセクタという単位で管理しており、１セクタは例えば５１２バイトである。このブロックデバイス層１０４でデバイスごとの入出力制御の違いと、ヘッドやシリンダなどパラメータの違いを吸収している。このように構成しているため、同時に複数の種類のデバイスを混在させることができる。

【００３０】本実施形態の電子カメラでは、特にブロックデバイス層１０４におけるフラッシュＲＯＭの記憶管理方法に特徴を有する。

【００３１】図１で示したフラッシュＲＯＭ１１には、現在いろいろなタイプのものがあるが、大きく分けてフラッシュＤＩＳＫ用に開発されたタイプとパーソナルコンピュータのＢＩＯＳ用に開発されたタイプがある。前者は消去単位がハードディスクで一般的な５１２バイトであり、ファイルシステムとの整合性が非常に良い。後

者のフラッシュＲＯＭは消去単位が例えば６４ｋなどといった大きなブロック単位でしか行えない様になっている。また、ＰＲＯＭの様に書き込み電圧として１２Ｖ等の電圧が必要なものもある。しかしながら、後者のタイプのフラッシュＲＯＭは安価で入手が容易である。本実施形態では、後者の様な特徴を持つフラッシュＲＯＭでありながらファイルシステムに対してハードディスク同様のサービスを提供する。

【００３２】＜フラッシュＲＯＭドライバインタフェース＞一般的にブロックデバイスがファイルシステムへ提供するサービスは以下の２つである。即ち、

（１）ロジカルセクタナンバーで指定したセクタからの読み出し

（２）ロジカルセクタナンバーで指定したセクタからの書き込み

である。そして、これに加えて

（３）ロジカルセクタナンバーで指定したセクタの開放の機能があれば、フラッシュＲＯＭのドライバは必要に応じて不要なセクタを消去することが可能となるため、効率良くフラッシュＲＯＭを消去することができる。

【００３３】（３）に挙げた機能は、通常のＤＩＳＫでは必要のない機能だが、キャッシュを持ったシステムだと、積極的にキャッシュリストから削除できるので、結果的にキャッシュのヒット率を上げる効果がある。ファイルシステムは、ファイルの消去等で不必要となったセクタを（３）の機能を用いてデバイスドライバへ通知する。フラッシュＲＯＭの消去は非常に時間がかかる処理だが、ＣＰＵ時間をほとんど消費しないためバックグランド処理で行うのが良い。

【００３４】後述の＜ＦＡＴキャッシュ＞においても説明するが、本実施形態のキャッシュは、新しいデータ（キャッシュ上に無いデータ）をアクセスする場合にキャッシュリストの中で最も古いデータを廃棄する。不要セクタをキャッシュリストの最後へ移動させる（即ち、最も古くアクセスしたデータがキャッシュの後ろへ移動する）ことで有効なデータがキャッシュから廃棄される可能性が低くなる。特にコンパイラー等の中間ファイルを多く生成するシステムでは、消去すべき中間ファイルがキャッシュに残っている可能性が高く、上記のキャッシュ管理はヒット率の向上に非常に有効である。

【００３５】図３は、デバイスドライバの管理ブロックをＣ言語で記述した宣言文を示す図である。構造体のＮｅｘｔは、次のデバイスへのリングポインタであり、メモリ中のデバイスを検索する目的で使用される。ＤｅｖＮａｍｅは、デバイスの名前として使用される。ＩｎｉｔＤｅｖは、デバイスの初期化ルーチンへのポインタである。ＳｈｕｔＤｏｗｎは、デバイスのシャットダウンルーチンへのポインタである。ＲｅａｄＳｅｃｔｏｒは、ロジカルセクタを指定して媒体の内容をバッファへ転送するルーチンへのポインタである。ＷｒｉｔｅＳｅｃｔｏｒは、ロジカルセクタを指定してバッファの内容を媒体へ転送する（書き込む）プログラムへのポインタである。ＲｅｌｅａｓｅＳｅｃｔｏｒはロジカルセクタを指定して、セクタを解放するルーチンへのポインタである。

【００３６】ファイルシステムは、この構造体を仲介してデバイスドライバを利用することになる。固定ディスクやフロッピーディスクの場合、ＲｅｌｅａｓｅＳｅｃｔｏｒには何も仕事をしないプログラムへのポインタが代入されている。または、ディスクキャッシュのキャッシュリストから指定セクタを削除するポインタでもよい。

【００３７】＜フラッシュＲＯＭ管理方法＞フラッシュＲＯＭに対するデータ書き込みは、上位層のファイルシステムからセクタ単位で行われる。図４は、フラッシュＲＯＭ上のセクタ構造の例を示す図である。図４において、１５１はイレースブロックである。このイレースブロック１５１は消去の単位であり、フラッシュＲＯＭの技術用語ではセクタと呼ばれるものである。しかしながら、ファイルシステムが扱う単位である“論理セクタ”と区別する為に、ここではイレースブロックと呼ぶことにする。

【００３８】図４によれば、システム中に複数のフラッシュＲＯＭ１５が搭載されていて、各フラッシュＲＯＭ１５は複数のイレースブロック１５１によって構成される。更に、各イレースブロック１５１は消去回数カウンタ１５２と複数のセクタ１５３によって構成されている様子をあらわしている。消去回数カウンタ１５２は、イレースブロック１５１を消去した回数をカウントする為に用いる部分である。各セクタ１５３は、管理領域とデータ領域とを有する。管理領域は、論理セクタ番号を表すセクタ番号１５４と、セクタが有効利用されているかどうかを表わす使用中フラグ１５５と、セクタとしての利用が終了したことを表わす使用済みフラグ１５６とで構成される。また、データ領域は、５１２バイトのデータ部１５７によって構成されている。

【００３９】データ領域と管理領域は、隣り合って配置する必要は無く、図５の様にまとめて管理することも考えられる。図５は、管理領域用データと、データ領域とを分離して格納する構成を表す図である。セクタ番号テーブルには、複数のセクタ１５３の各セクタ番号１５４が格納される。また、フラグテーブルには、使用中フラグ１５５、使用済みフラグ１５６が格納される。更に、データテーブルには、データ部１５７の内容が格納される。以上のようなデータ構成をとることも可能であるが、少なくとも管理領域と、これに対応するデータ領域とは同じイレースブロック内に納めるのが好ましい。

【００４０】なお、システムは「使用中フラグ１５５」より「使用済みフラグ１５６」の方を優先的に評価する。図６は、各フラグの状態に対応した意味を示す図で

ある。図中、ＦＡＬＳＥは、消去後の状態と同じ値をとる。使用中フラグ１５５がＴＲＵＥであっても、使用済みフラグ１５６がＦＡＬＳＥであれば、当該セクタのデータは無効である。

【００４１】＜フラッシュＲＯＭの論理セクタ書き換え＞フラッシュＲＯＭはＰＲＯＭ同様、データを書き換える為に、一度消去してから再書き込みをしなければならない。しかも、消去の最少単位が大きく（例えば６４ｋバイト）消去時間が長い（例えば１秒）。そこで上位層のファイルシステムが、特定のセクタを書き換えようとした場合、消去済みの領域へ論理セクタを移動させることで、消去動作をせずに、見かけ上で論理セクタのデータ書き換えを実現する。

【００４２】図７はセクタの書き換え手順を説明する図である。同図を用いて、８番セクタ（論理セクタ番号が８のセクタ）の書き換えを例にして詳しく説明する。図７中、左側が書き換え前の状態であり（（ａ）の状態）、右側が書き換え後の状態（（ｂ）の状態）である。また、図７において、管理領域中の数字は論理セクタ番号を表し、（使用中）は使用中フラグ１５５がＴＲＵＥで使用済みフラグ１５６がＦＡＬＳＥの状態、（使用済）は使用中フラグ１５５と使用済みフラグ１５６が共にＴＲＵＥの状態を示す。

【００４３】“セクタ番号８（使用中）”の場所に、８番セクタのデータが格納されている。今、８番セクタがＦＡＴやファイルの一部として利用されていて、その内容を変更したい場合に上位層から８番セクタの書き換え要求が発生したとする。書き換え要求が発生すると、フラッシュＲＯＭのデバイスドライバは、フラッシュＲＯＭの未使用セクタを検索し、その場所を新たな８番セクタの場所としてセクタ番号と更新後のデータを格納し、使用中フラグをＴＲＵＥにする。次に、以前８番セクタだったセクタの使用済みフラグをＴＲＵＥにする。このような手順で８番セクタのデータの書き換えが実現される。

【００４４】＜ガベージコレクション＞以上の様な方法で論理セクタの書き換えを実行していくと、いずれフラッシュＲＯＭのほとんどの領域を“使用済セクタ”にしてしまうことになる。そこであるタイミングでフラッシュＲＯＭを一旦消去して“使用済セクタ”を“未使用セクタ”へ戻す必要がある。基本的なガベージコレクションの動作を図８を用いて説明する。図８は、本実施形態におけるフラッシュＲＯＭのガベージコレクション動作を説明する図である。

【００４５】図中（Ａ）は、ガベージコレクション前の状態である。説明を簡単にするために、本例のフラッシュＲＯＭはセクタ６個分の大きさのイレースブロックで構成されているものとする。イレースブロック（１）には使用済セクタが３個と使用中セクタが３個あり、消去回数は５回である（消去回数カウンタ１５２の内容が５である）。イレースブロック（２）には使用中セクタが１個、使用済セクタが１個、未使用セクタが４個あり、消去回数は９回である。この状態からガベージコレクションを開始する。

【００４６】先ず、調整対象イレースブロックを選定する。ここで調整対象イレースブロックは消去を行う対象としてイレースブロックとなる。整理対象イレースブロックの選定は、使用済セクタをたくさん含むイレースブロックから優先的に選択すると整理効率が良い。しかし使用済セクタを含まない場合を別として消去回数の少ないイレースブロックを優先的に整理対象とする方法を取ればチップ内のイレースブロックを平均的に使用することができ、書き換え耐久を分散させることができる。選択手順の詳細についてはフローチャートを用いて後述する。

【００４７】今、イレースブロック（１）が整理対象として選定されたとする。次に、整理対照であるイレースブロック（１）の使用中セクタ（使用中フラグがＴＲＵＥで、使用済みフラグがＦＡＬＳＥのセクタ）を他のイレースブロックに移動させる。使用中セクタの移動手順は、セクタの書き換え時と同様に、他のイレースブロック中の未使用セクタを検索して、使用中セクタ内のデータ領域と管理領域の内容をコピーし、移動元の使用中セクタの使用済みフラグをＴＲＵＥにする。なお、未使用セクタが無い場合の処理は、後で述べる。

【００４８】図８の（Ｂ）は、イレースブロック（１）の使用中セクタをすべてイレースブロック（２）へ移動させた状態である。この結果、イレースブロック（１）には、使用済セクタしか存在しないことになる。

【００４９】次に、使用済セクタだけで構成されているイレースブロックを検索する。ここで、検索を行うのは、通常の書き換え動作の際に偶然イレースブロック内のセクタが全て使用済セクタとなっている場合があるからである。続いて検索されたイレースブロックに対して消去を行う。消去には時間がかかるが、複数のイレースブロックを同時に消去できるため、できるだけ一度に複数のイレースブロックを消去するのが良い。消去が終了すると消去回数カウンタへ消去前の値＋１したものを書く。これでガベージコレクション完了である。

【００５０】図８の（Ｃ）がガベージコレクション終了時の状態である。イレースブロックをできるだけ同時に消去した方が効率が良いため、使用済セクタと未使用セクタがある限りたくさんのイレースブロックを同時に調整すると良い。極端に消去回数カウンタの値が他のイレースブロックより少ないものがあれば、使用済セクタを含んでいなくても整理さえすれば、書き換え耐久の分散を図れる。また、一度整理するとデータの配列が変わる為、書き換え耐久分散のきっかけとなる。

【００５１】＜未使用セクタがない場合＞次にシステム中に使用済セクタが有るにもかかわらず未使用セクタが

全く無くなってしまった場合のガベージコレクション手順を図９を使って説明する。図９は、未使用セクタが存在しない場合のガベージコレクションの動作を説明する図である。

【００５２】先ず、上述した基本的なガベージコレクション手順に従い、イレースブロック（１）を整理対象として選択する。次にイレースブロック（１）の使用中セクタを移動する為の未使用セクタを検索する。未使用セクタがある場合は、上述の基本的ガベージコレクションと同様にセクタの移動を行う。

【００５３】一方、検索の結果、未使用セクタが無ければ、ＤＲＡＭ１４のヒープエリアからデータの退避に必要な大きさのメモリブロックをアロケーションする。そして調整対象イレースブロック内の使用中セクタをＤＲＡＭ１４へコピーする。この場合は、フラッシュＲＯＭの別の領域へセクタを移動する場合と違い、元のセクタの使用済フラグをＴＲＵＥにしない。なぜならこの時点で電子カメラの電池７が外れるなどの事故が起こった場合に、ＤＲＡＭ内のデータが消滅してしまい、データの修復ができなくなるからである。図９の（Ｂ）は、ＤＲＡＭ１４の領域へコピーされたセクタを表現している。調整対象のイレースブロックの使用中セクタをすべて退避出来たら、その調整対象のイレースブロックを選択して消去する（図９の（Ｃ）参照）。消去が終わった後は未使用セクタがたくさん出来ているはずである。よって、次に未使用領域を検索してＤＲＡＭ１４へ待避してあったデータを復元する。図９の（Ｄ）は、ガベージコレクションが完了した状態を示している。

【００５４】上記の手順でガベージコレクションをした場合でも、イレースブロックを消去してからデータを復元するまでの間に電子カメラの電池７が外れるなどの事故が起こったらデータの修復をすることはできない。つまり、セクタのデータをＤＲＡＭ１４に退避する方法はできる限り取らない方が、よりシステムの安全性を保つことができる。一回、ＤＲＡＭを使ってガベージコレクションを行えば未使用セクタができる。したがって、１度ＤＲＡＭ１４を使ったガベージコレクションを行い、その後に通常のガベージコレクションを行えば、消去時間は余分にかかるが安全性を高めることができる。逆にＤＲＡＭ１４への退避を積極的に行う（例えばヒープ領域がある限り退避する）と、同時に整理できるイレースブロックが増える為効率を上げることができる。従って、安全性と効率のどちらを優先するかを指定できるように構成してもよい。またシステムの電源が電池７より供給されている場合は安全性優先、ＡＣアダプタから供給されている場合は効率優先に自動的に切り替わるように構成してもよい。本処理については図３６を参照して後述する。

【００５５】＜イレースブロックを余分に１つ用意＞残り容量が極端に少なくなるとガベージコレクションが多発してシステムのパフォーマンスが極端に落ちる。総論理セクタ分を格納できるイレースブロック数よりも１つだけ余分にイレースブロックを使用すれば、そのような事態を避けることが可能である。仮に、１イレースブロックあたり１２７セクタ格納できるとして、全てのセクタが使用中となった場合、同じ１セクタを１０回書換える場合を例にすると、余分イレースブロックがなければ、１０回の消去と１２７０セクタの書き込みが発生する。しかし、イレースブロックを余分に１つ用意しておけば、１０セクタの書き込みしか発生しない。

【００５６】よって、本実施形態では、イレースブロックの数はチップの構成で決まるので、最低１つのイレースブロックが余る様な総論理セクタ数を設計する。

【００５７】＜ガベージコレクションのタイミング＞ガベージコレクションは消去動作を伴うために非常に時間がかかる。そのためガベージコレクションをいつ行うかによってカメラの使い勝ってを左右することとなる。例えば、セルフタイマなどの数秒間撮影しなくても良い時にガベージコレクションを行えばユーザがストレスを感じることがない。

【００５８】＜ＲＡＭ上の記憶場所管理＞フラッシュＲＯＭ１５上ではセクタ番号と実際の記憶場所が関連していないために特定のセクタを読み書きする為にフラッシュＲＯＭ１５を検索しなければならない。そこでシステムがリブートする際に、フラッシュＲＯＭ１５における各セクタの格納アドレスを示す記憶場所管理テーブルをＤＲＡＭ１１４上に作成しておくと、フラッシュＲＯＭ１５に対して高速なデータの読み書きを実現できる。一度、記憶場所管理テーブルを作成すれば、フラッシュＲＯＭ１５に対するセクタの書き込みやガベージコレクションによって記憶場所に変更が生じた場合に限って記憶場所管理テーブルの記憶位置を更新するだけで常に正しい記憶場所管理テーブルを維持することが可能である。

【００５９】図１０は、ＤＲＡＭ上に作成された記憶場所管理テーブルを説明する図である。図中、右側にＲＡＭ上の作成した記憶場所管理テーブル１４０を示した。０セクタと４セクタは記憶場所不在を意味する値（ＮＵＬＬ）が入っている。これらは、フォーマット後そのセクタに対する書き込みが全く無かったか、もしくは、ファイルシステムが開放したセクタである。

【００６０】ファイルシステムがファイルの消去などで不要となったセクタを解放する命令をドライバに出した場合のデバイスドライバの動作は次のようになる。まず、ＤＲＡＭ１４上の記憶場所管理テーブル１４０の指定されたセクタのポインタを参照してフラッシュＲＯＭ１５上の現在使用中の該当するセクタを探し出す。そして、当該セクタの使用済フラグをＴＲＵＥにし、ＤＲＡＭ１４上の記憶場所管理テーブル１４０の指定セクタのポインタへ不在値（ＮＵＬＬ）を代入する。

【００６１】なお、ガベージコレクションの為にセクタ

の内容をＤＲＡＭ１４へ待避している場合は、記憶場所としてＤＲＡＭへのポインタが代入されている。また、同一論理セクタに対する同時操作を禁止する為のロック変数もテーブルに納めることが望ましい。

【００６２】＜ＭＳ－ＤＯＳのファイル復元＞本電子カメラとパーソナルコンピュータ２２で、記憶媒体上のデータ交換が出来ると都合が良い。本実施形態で説明したフラッシュＲＯＭ管理方式を使用して、現在パーソナルコンピュータで普及しているＭＳ－ＤＯＳ（商標）と互換性があるファイルシステムを実装することができる。ＭＳ－ＤＯＳには一度消去したファイルを復元するユーティリティが付属している。ところが、本実施形態ではフラッシュＲＯＭの消去効率を向上させる為に、消去したセクタのデータ部を失ってしまう様な構成となっている。カメラで消去した媒体をパーソナルコンピュータで復元する事が原理的にできない構成になっているのである。

【００６３】パーソナルコンピュータでのファイル復元機能を禁止できれば、このような事故を防ぐことができる。本実施形態では、ＭＳ－ＤＯＳがファイル復元の時に使用するデータを破壊することでファイル復元機能を禁止する。これをいかに説明する。

【００６４】ＭＳ－ＤＯＳ（商標）で、ファイルを消去するとディレクトリに空きスロットができる。ディレクトリにはファイル名／タイムスタンプ／最初のクラスタなどの情報が格納されている。図４５はディレクトリスロットの特徴を表す図である。ディレクトリスロットの最後には、リストの最後であることを示すEndOfDirが格納されている。

【００６５】今、File Bを削除すると、ファイル名の先頭が削除を表すシンボルに置き換えられ、ＦＡＴのクラスタチェーンが消去される。この様子を図４６に示す。

【００６６】アンデリートプログラムは、２番目のスロットに残った情報を元に、ファイルの復元を試みる。逆にこの情報がなければ、ファイルの復元を防止できる。

【００６７】図４７は本実施形態のＤＯＳ互換ファイルシステムでファイルを消去した後の状態を表している。本実施形態では、ディレクトリエントリテーブルの最後に格納されているファイルを消去したいファイルのエントリに上書きし、ディレクトリエントリテーブルの最後のファイルだった部分にEndOfDirを上書きするように構成する。こうすることにより、ファイル復元機能によるファイルの復元を防止できる。

【００６８】なお、ファイルの消去時にはＭＳ－ＤＯＳと同様にセクタのデータをそのまま残しておき（セクタの開放を行わない）、ガベージコレクション時にまとめてＦＡＴとデータの関係を参照してしながら不要部分を消去する方法もある。

【００６９】＜バックグランドで前処理＞あるフラッシュＲＯＭでは、消去前のデータが“０”になっている方が高速に消去処理できる。フラッシュＲＯＭの消去完了の確認は、データ書き込み時と同様にデータポーリングによって行われる。従って、このようなフラッシュＲＯＭを使う場合は、バックグランド処理で“使用済”となったセクタのデータを０に書き換える「前処理」を行うことで性能を向上させることができる。最も低いプライオリティのタスクとして実行するようにしておけば、スループットの低下にはつながらない。

【００７０】この前処理バックグラウンドでの“前処理済セクタ”管理の為にフラグを用意しておけば前処理の効率を上げることができる。

【００７１】そのために、セクタのとりうる状態として、「未使用」「使用中」「使用済」に加えて「前処理済」の４つの状態を表示できる管理フラグをフラッシュＲＯＭのセクタ内部へ用意すると効率が良い。

【００７２】図３８は、本実施形態における消去処理速度向上のための前処理の制御手順を表すフローチャートである。同図において、ステップＳ２５０１にて、使用済でかつ前処理の済んでいないセクタを抽出する。これは、セクタ内の管理フラグが「使用済」となっていて、かつ「前処理済」となっていないセクタを抽出することで実現できる。ステップＳ２５０２において、抽出されたセクタに対してデータ「０」を上書きを開始する。ステップＳ２５０３では当該セクタについて前処理を終了したか否かを判断する。フラッシュＲＯＭへの書込みは１バイト単位であるので、１セクタ分のバイト数の書込みが必要となる。当該セクタに対する前処理が終了していなければステップＳ２５０４へ進み、他のタスクへ制御を移す。

【００７３】上述したように本処理は最もプライオリティの低いタスクで行われるので、ＣＰＵ５がアイドル状態となったときに再び本処理が実行される。この場合処理はステップＳ２５０３へ戻る。この時点で、前回の書込みが終了していなければそのまま他のタスクへ処理を移行する。

【００７４】以上のようにして当該セクタの全バイトに対して「０」の書込みを終えると、ステップＳ２５０３からステップＳ２５０５へ進み、当該セクタの管理フラグを、「前処理済」を示す状態にセットする。そして、引き続き、他のセクタについて前処理を行うために、ステップＳ２５０１へ戻る。

【００７５】＜ＦＡＴキャッシュ＞本システムでは、書き込み発生の度に記憶場所を変更し、その度に「未使用セクタ」が発生する。そこで、使用頻度の多い部分を特に優先的にバッファリングするキャッシュが有ればトータルの書き込み頻度が激減する事が予想される。キャッシュとして用意するメモリは多ければ多いほど良いが、システムのメモリには限界がある。

【００７６】本来使用頻度の高いセクタのデータは、キャッシュ中に存在する確率も高いが、使用頻度の低いセ

クタを大量に読み書きした場合、当然キャッシュから吐き出されることになる。

【００７７】そこで、ファイルシステムが管理する管理領域を優先的にキャッシングする様に構成すれば、スループットの向上を期待できる。なぜならファイルシステムの管理領域は頻繁に更新されているからである。

【００７８】パーソナルコンピュータで普及しているＭＳ－ＤＯＳのＦＡＴシステムの場合、７２０ｋや１．４Ｍといったフォーマット形式では１クラスタが１セクタで構成されている為、シーケンシャルにファイルを読む場合でも２回に１回はＦＡＴを読まなければならない。ファイルを書く場合は、さらにたくさんのＦＡＴアクセスが発生する。このため、システム中にたくさんのファイルがオープンされるとキャッシュのヒット率が落ちてしまう。

【００７９】アプリケーションソフトウェアにもよるが、ＦＡＴシステムにおいてＦＡＴのみを対処にしたキャッシュは、ＤＩＳＫ全体を対象にしたキャッシュに対して１／２のメモリで同等のヒット率を確保できる。図１１はキャッシュソフトウエアの階層的な位置付けを表す図である。キャッシュのソフトウェアは図１１の様にファイルシステムとフラッシュＲＯＭの中間的な場所となる。

【００８０】図１２はキャッシュの主記憶上のデータ構造を表わす図である。片方向線形リスト構造でバッファ全体を管理している。検索方向順にデータが古くなっている。論理セクタ番号が１２，１１，６，５の順番でアクセスすれば、図１２に示されるような順番となる。また、各セクタには、変更フラグが設けられており、キャッシュ上でデータの更新があった場合、変更フラグがＦＡＬＳＥからＴＲＵＥに変化する。このようなＦＡＴキャッシュの読み出し手順、及び書き込み手順を図１３、１４を参照して説明する。図１３はＦＡＴキャッシュの読み出し手順を表すフローチャートである。図１４はＦＡＴキャッシュの書込み手順を表すフローチャートである。

【００８１】図１３において、ステップＳ１５０１でＮセクタの読み出しを開始する。ステップＳ１５０２でＮセクタがＦＡＴかどうかを判断する。ＦＡＴでなければ、ステップＳ１５０９でフラッシュＲＯＭ１５からデータを読み出す。

【００８２】一方、ステップＳ１５０２でＮセクタがＦＡＴならステップＳ１５０３へ進み、キャッシュリストを検索する。ここでは、図１２で説明した片方向線形リストを検索することになる。キャッシュ中にＮセクタが存在すればステップＳ１５０７へ進み、Ｎセクタのバッファからデータを読み出す。

【００８３】また、ステップＳ１５０３でＮセクタがキャッシュリスト中に存在しなかった場合は、ステップＳ１５０４へ分岐し、最も長くアクセスされていないセクタのデータ（図１２ではセクタ番号１２のデータ）の吐き出しを行う。まず、ステップＳ１５０４では、キャッシュリストの最後の項の変更フラグを判断する。もし変更フラグがＴＲＵＥなら、ステップＳ１５０５へ進み、変更内容をフラッシュＲＯＭ１５へ書き込む。変更が無い（変更フラグがＦＡＬＳＥの場合）なら、そのままステップＳ１５０６へ制御を移す。読み出し手順の中で書き込みを行うのは奇妙に思うかもしれないが、バッファがキャッシュの吐き出しが起こるまで極力書き込み動作を行わない方が効率が良い。

【００８４】ステップＳ１５０６でフラッシュＲＯＭ１５からリスト最後のバッファへＮセクタの内容を読み出す。ステップＳ１５０７でＮセクタのバッファからデータを読み出す。ステップＳ１５０８でＮセクタのバッファをキャッシュリストの先頭へ移動させる。これは、図１２において、各セクタが有する「次のバッファー」（次のバッファーを示すアドレス）の値を変更することで達成される。ＦＡＴキャッシュへのアクセスが行われる度にステップＳ１５０８の動作が繰り返されることで、自然にアクセスされないバッファがリストの先頭から最後に向かってシフトしていく。よって、ステップＳ１５０４でリスト最後のバッファを選ぶのは、最も古いバッファを吐き出す為である。

【００８５】次に図１４を参照して書込み手順を説明する。

【００８６】ステップＳ１６００でＮセクタの書き込みを開始する。ステップＳ１６０１では、ＮセクタがＦＡＴかどうかを判断する。ＦＡＴでなければ、ステップＳ１６０８へ進み、フラッシュＲＯＭ１５へのデータの書き込みを実行する。

【００８７】一方、ステップＳ１６０１でＮセクタがＦＡＴならば、ステップＳ１６０２へ進み、キャッシュリストを検索する。キャッシュ中にＮセクタが存在すればステップＳ１６０６へ進み、Ｎセクタのバッファへデータの書き込みを実行する。

【００８８】また、ステップＳ１６０２でＮセクタがキャッシュリスト中に存在しなかった場合は、ステップＳ１６０３へ分岐し、バッファから最も長くアクセスされていないセクタの吐き出しを行うとともに、Ｎセクタをキャッシュに登録する。まず、ステップＳ１６０３でキャッシュリストの最後の項の変更フラグを判断する。もし変更フラグがＴＲＵＥなら、ステップＳ１６０４で変更内容をフラッシュＲＯＭへ書き込み、ステップＳ１６０５へ進む。また、変更が無いなら（変更フラグがＦＡＬＳＥなら）そのままステップＳ１６０５へ制御を移す。ステップＳ１６０５では、キャッシュリストの最後の項をＮセクタとする。その後、ステップＳ１６０６で、Ｎセクタのバッファへデータの書込みを実行する。

【００８９】その後、ステップS1607でN、セクタのバッファをキャッシュリストの先頭へ移動させる。書

き込み手順の中でフラッシュＲＯＭへの書き込みを行わないのは奇妙に思うかもしれないが、キャッシュの吐き出しが起こるまで極力書き込み動作を行わない方が効率が良い。

【００９０】また、ステップＳ１５０１及びステップＳ１６０１における、ＦＡＴの判断であるが、ＩＣカード等の完全に上位層（ファイルシステム）の情報を共有できないシステムでも、書き込みデータの内容を解析することでＦＡＴ領域の場所を特定できる。なぜならば、論理セクタ０に相当する部分にＦＡＴの位置等の情報か格納されていることが決まっているからである。

【００９１】＜フラッシュＲＯＭへの１バイトの書き込み＞フラッシュＲＯＭ１５に対する全ての（管理領域を含む）読み書きは、最終的に１バイトの読み書き命令によって実行される。フラッシュＲＯＭ１５の書き込みには、通常のＰＲＯＭ同様の時間がかかる。１バイトの書き込みが終了するまでは、同じチップへの書き込みはできない。書き込み終了信号として信号線が用意されているチップと特別な信号が用意されていないチップがある。後者の場合は、データポーリングと言う手法で書き込み終了を確認しなければならない。データポーリングとは、ベリファイに非常によく似た方法で、書き込みデータと読み出しデータが一致するまで待つビジー制御方法である。

【００９２】信号線によって書き込み終了を知ることが出来る場合は、ＣＰＵ５への割り込みと併用して書き込み待ち中のＣＰＵタイムを別のタスクへ割り当てることができる。

【００９３】上述のように、信号線が無いチップの場合は、データポーリングを行なわなければならない。データ書き込みの効率をあげる為にはいくつものチップに対してパイプライン的に書き込みを行い、データポーリング時間のロスを押さえなければならない。そのため、１バイトの書き込みが完了する前に次の動作へ制御を移す必要がある。新たな読み書きを行う前に以前の書き込みが完了しているかどうかを確認するのが良い。図１５は、その様子をＣ言語で表現したものである。

【００９４】図１５の１行目は、データ書き込みを行う関数の入り口である。最初の引数は最後に書き込んだアドレスとデータを保存するための構造体へのポインタ、第２の引数は書き込むアドレス、第２の引数は書き込むデータである。

【００９５】３行目では、最後に書き込んだアドレスを参照してチップに書かれたデータと最後に書いたデータを比較して、両者が一致するまでループを実行する。これがデータポーリングである。前回の書き込みが完了するとこのループから抜け出す。

【００９６】４行目で新しいアドレスへＤａｔａを書き込む。５行目と６行目で、今回書いたアドレスとデータを保存する。この情報は、次のデータポーリングで利用される。

【００９７】リスト７行目のRotateRdyQueueは、自タスクの次に実行されるべき同一プライオリティの実行可能状態のタスクへＣＰＵを譲るオペレーティングシステムのシステムコールである。

【００９８】９行目は読み出し関数の入り口である。第１の引数はアドレスとデータを保存するための構造体へのポインタ、第２の引数は読み出すアドレスである。この関数は上位のプログラムに対して第２の引数で指定されたアドレスに格納されたデータを返す。

【００９９】１１行目では、もし読み出そうとしたアドレスが最後に書き込んだアドレスなら戻す値は最後に書き込んだデータなので構造体の中に保存された情報を返す。１２行目は３行目と同じようなデータポーリングである。データポーリングに成功しないと同じチップの別のアドレスを読むことができない。データポーリングが終わって１３行目で指定したアドレスの内容を戻している。

【０１００】１チップへの書き込みを以上の様な構成にしておけば、チップ数と書き込みタスクを増やすだけで確実に見かけ上の書き込み速度を向上させることができる。また、全体のスループットを上げる為にわざとチップ数分のセクタバッファを用意（２チップなら２セクタ）して書き込む内容がバッファに溜まるまで処理しないようにすると効果がある。

【０１０１】図１５のプログラムの特徴的なところは、データポーリングをデータ書き込み直後に行うのではなく、次の書き込みの前に行うことである。そのために前回書いたアドレスとデータを保存しておくＲＡＭ領域をチップごとに確保し構造体「struct DEV」として格納しているのである。

【０１０２】図３９は、本実施形態におけるフラッシュＲＯＭへの１バイトデータの書込み手順を表すフローチャートである。本フローチャートは、１つのフラッシュＲＯＭチップへの書込みの制御手順を示している。ステップＳ２６０１では、前回の書込み処理が完了したか否かを判断する。前回の書込み処理が終了していなければステップＳ２６０４へ進み、そのまま他のタスクへ制御を移す。

【０１０３】一方、前回の書込み処理が終了していれば、次の書込みデータを準備し、これをＤＲＡＭ１４へ保存する。上述のステップＳ２６０１における書込み終了の判断は、フラッシュＲＯＭに書き込まれたデータと、このステップＳ２６０２で保持されたデータとの比較によって行われる。

【０１０４】続いて、ステップＳ２６０３において、データの書込みを開始する。以上のような処理によれば、複数のフラッシュＲＯＭチップに対して、複数のタスクで書き込みを行うような場合に、いわゆるラウンドロビン方式を適用した書き込み処理が可能となり、複数のＲ

ＯＭチップに対して効率良くデータの書き込みが行える。なお、マルチタスクの管理プログラムは、上述のＲＯＭ１３に格納されている。組み込みようのリアルタイムＯＳとしては、VxWorks（商標）やpSOS（商標）等が市販されており、ＲＯＭ１３にこれらのようなリアルタイムＯＳが格納されている。

【０１０５】＜フラッシュＲＯＭ書き込み電源の共有化＞データの書き込みや消去の際にＰＲＯＭ同様に１２Ｖ等の特別な書き込み電圧を必要とするチップや、書き込み電圧を与えることで書き込みが高速になるチップがある。この様なチップを使用する場合に専用のＤＣ／ＤＣコンバータ等の電圧発生部を設けると電子カメラのコストアップにつながる。ところが、従来よりカメラにはストロボの充電や、機構部分やＣＣＤの駆動等、特別な電圧が必要な部分がありＤＣ／ＤＣコンバータ等を搭載している。そこで、フラッシュＲＯＭの書き込み電圧とストロボ充電やメカ駆動を時分割多重で行うことで、少容量のＤＣ／ＤＣコンバータでシステムを構築でき、システムのコストを押さえることができる。

【０１０６】図１６は、ＤＣ／ＤＣコンバータの出力容量を越えない様に電源を管理するプログラムをＣ言語で表現したものである。Ｌｉｎｅ１～６が１ステップのズームアップ関数で、Ｌｉｎｅ７～１３がフラッシュＲＯＭへ１セクタ書き込む書き込み関数である。ズームアップ関数はＬｉｎｅ３でＤＣ／ＤＣコンバータの資源管理用のセマフォ“ＳｅｍＤＣＤＣ”を獲得して、モータを１ステップ動かす関数をＬｉｎｅ４で呼び出す。モータ駆動が終わるとＤＣ／ＤＣコンバータの資源管理用のセマフォ“ＳｅｍＤＣＤＣ”を開放する。セマフォはマルチタスクのオペレーティングシステムで資源を管理する為の一般的な方法であり、多くのオペレーティングシステムがシステムコールとして用意している。

【０１０７】即ち、Ｌｉｎｅ３で既に“ＳｅｍＤＣＤＣ”が他のタスクによって使用されていたとすると、他のタスクがセマフォ“ＳｅｍＤＣＤＣ”を開放するまでズームアップをしようとしたタスクの実行が保留される。

【０１０８】書き込み関数はＬｉｎｅ９でセマフォ“ＳｅｍＤＣＤＣ”を獲得し、フラッシュＲＯＭへ１セクタのデータを書き込む。Ｌｉｎｅ１１でデータポーリングを行い最後の書き込みが終了したことを確認したら、Ｌｉｎｅ１２でセマフォ“ＳｅｍＤＣＤＣ”を開放する。このようにプログラムを構成すれば、ズームアップとフラッシュＲＯＭの書き込みを同時に行うことは無くなる。ズームは１ステップ単位であり、書き込みはセクタ単位なので非常に短い保留時間の後に必ず電源を獲得できる。

【０１０９】図１６について更に説明すると、図１６のＬｉｎｅ１はズームアップする関数の入り口である。本関数には引数はない。Ｌｉｎｅ３で電源の使用権利として宣言したSemDCDCの権利を一つ獲得する。この時、使用権利が１つもなければこの関数を呼び出したタスクの実行は保留される。電源の使用権利を別のタスクが解放すればZoomUpを呼び出したタスクが再び実行可能状態に戻る。そして、Ｌｉｎｅ４のモーターを動かす関数を呼び出すことができる。そして、Ｌｉｎｅ５で、電源使用権利を返却してこの関数の仕事は終了する。Ｌｉｎｅ７は１セクタのデータをＥＥＰＲＯＭに書き込む関数の入り口であり、Ｌｉｎｅ９ｄｅ電源利用権利を獲得してＬｉｎｅ１２で返却している。

【０１１０】図４０は、上述した電源の共有手順を説明するためのフローチャートである。同図において、ステップＳ１７０１で、電源コントローラ９によるＤＣ／ＤＣコンバータ８の出力電力の供給が解放されたか否かを判断する。ステップＳ１７０２では、電源確保のための指示の内容を解析し、この指示結果に従って、ステップＳ１７０３、１７０５、１７０７、１７０９のいずれかに分岐する。

【０１１１】指示の内容が、ＣＣＤ駆動電力の供給であれば、ステップＳ１７０３へ進み、ＣＣＤ２に対してＣＣＤ駆動のための電力を供給する。そして、ステップＳ１７０４にて、ＣＣＤ駆動の終了（即ち撮影動作の終了）を検出すると、ステップＳ１７１１へ進み、電源の解放を行う。また、ストロボの充電要求であれば、ステップＳ１７０５へ進み、電源コントローラ９に対してストロボ２１に対する充電電力を提供させる。そして、ステップＳ１７０６でストロボの充電を完了したら、ステップＳ１７１１へ進み、電源の解放を行う。なお、充電の電力供給は、所定時間の充電を行う毎に他の電源供給のために電源を解放する。即ち、ストロボ２１への充電を管理するプログラムは別個に所定のタスクに存在し、充電の完了はそのタスクによって管理される。

【０１１２】指示の内容が、ズーム機構の駆動であれば、ステップＳ１７０７へ進み、ズーム機構の駆動系（不図示）へ電力供給を行う。そしてステップＳ１７０８で、１ステップのズーム動作を終えたらステップＳ１７１１へ進み、電源を解放する。更に、指示の内容がフラッシュＲＯＭへの書込みであれば、ステップＳ１７０９へ進み、フラッシュＲＯＭ１５への書込み電力を供給する。１セクタ分の書込みが終えたら、ステップＳ１７１０からステップＳ１７１１へ進み、電源を解放する。

【０１１３】なお、ステップＳ１７０４、１７０６、１７０８、１７１０において、各動作の終了を待つが、この待ちループにおいて、他のタスクへの制御が移り、マルチタスク処理が遂行される。この管理処理は、各タスクから随時起動が可能であり、複数のタスクで同時に起動される可能性もある為、ステップＳ１７０１で電源解放のチェックを行っている。

【０１１４】以上の図４０のフローチャートによれば時分割で電源を利用することが可能となる。しかしなが

ら、すべてのシステム（ＣＣＤ／ストロボ／ズーム／フラッシュＲＯＭ）が依存しあった１つのプログラムである。このようなソフトウエアを開発すると、開発／デバッグ／メンテナンスのコストが大きくなり、拡張性や柔軟性を保つのが難しくなる。

【０１１５】そこで、電源を１つの資源に見立ててＯＳの提供する資源管理機能を用いることで開発効率を向上させることができる。そこで上述のセマフォによる資源管理を行う。即ち、ＣＣＤの駆動部、ストロボの駆動部、ズームの駆動部、フラッシュＲＯＭの駆動部のそれぞれの制御プログラムが、電源という資源（セマフォ）を獲得、解放することで、時分割された電源の割当てが行える。

【０１１６】図４８は本実施形態による電源の時分割利用を説明する図である。同図に示されるように、あるタスクＡ（例えばＣＣＤ）によって電源要求が発生したとき、電源セマフォが解放された状態にあれば、そのセマフォを獲得して、電源を占有する（ステップＳ２００１～Ｓ２００３）。続いてステップＳ２００４において、当該電源よりの電力供給を得て所定の処理を行うと、ステップＳ２００５へ進んでセマフォを解放する。

【０１１７】一方タスクＡより遅れて電源獲得を要求したタスクＢでは、ステップＳ２０１１における電源要求ではセマフォを獲得できず、ステップＳ２０１２により、セマフォの解放待ちとなる。そして、タスクＡよりセマフォが解放されると、このセマフォをタスクＢが獲得して、電源を占有する（ステップＳ２０１３）。その後タスクＢで所定の処理を実行し（ステップＳ２０１４）、電源を解放する（ステップＳ２０１５）。

【０１１８】以上のようなセマフォによる電源資源の管理により、電源の時分割利用が可能となる。

【０１１９】なお、図４８によれば、電源資源の利用権利を示すセマフォが一つしかないが、複数個のセマフォが存在するようにしても良いことは言うまでもない。

【０１２０】＜実施形態の電子カメラの動作説明＞図１７は、本実施形態のリブートからサービスの開始までの動作手順を表わすフローチャートである。ステップＳ１０１でシステムがリブートすると、ステップＳ１０２でフラッシュＲＯＭ１５の管理領域をスキャンし、ＤＲＡＭ１４上に記憶場所管理テーブル１４０を作成する。また、この処理と並行して、ＤＲＡＭ１４上の未使用セクタカウンタ、使用済セクタカウンタ、使用中セクタカウンタへ、それぞれの状態に対応するセクタがいくつ有るかを数え、セットする。このカウンタは、後にフラッシュＲＯＭ１５に対して操作を行ったときに更新され、記憶効率を判断するのに用いられる。その後、ステップＳ１０３へ進み、各種のサービスを開始する。

【０１２１】図１８は、指定セクタの読み出しサービスの手順を表わすフローチャートである。まず、ステップＳ２０１でＮセクタの読み出しを開始する。ステップＳ２０２では、Ｎセクタをロックする。セクタのロックはロック変数を使って行う。このロック変数は、記憶場所管理テーブル１４０で各セクタの記憶場所とともに管理される。ステップＳ２０２では、セクタが既に他のタスクによってロックされている場合、他のタスクによってアンロックされるのを待ち、他のタスクによってアンロックされた後で当該セクタのロックを行う。ロックしたセクタはステップＳ２０６でアンロックするまでの間、自タスクによって占有することが出来る。

【０１２２】ステップＳ２０２で論理セクタをロックすると、ステップＳ２０３で記憶場所管理テーブルを参照して、当該セクタに有効なデータ記憶されているかどうかを確認する。有効なデータが記録されて無い場合は、ステップＳ２０４へ分岐する。ステップＳ２０４では、ダミーのデータ（例えば全部０など）をセクタの内容として読み出す。ステップＳ２０３で有効なデータが格納されていると判断された場合は、ステップＳ２０５へ分岐する。ステップＳ２０５では記録場所管理テーブルの値を元にフラッシュＲＯＭ（または主記憶）からデータを読み出す。

【０１２３】ここで、ガベージコレクションを実行中でＮセクタが主記憶（ＤＲＡＭ１４）へ退避されていた場合は、記憶場所管理テーブルのポインタは主記憶をポインティングとしている。また、図中点線で囲んだ部分はＮセクタを占有している期間である。この様なロック機構によって１つのセクタ操作の安全性を保証している為、ガベージコレクションの途中でも操作中でないセクタを自由に読み出すことが可能となっている。

【０１２４】図１９は論理セクタの書き込みサービスの手順を表わすフローチャートである。ステップＳ３０１でＮセクタの書き込みを開始する。ステップＳ３０２でステップＳ２０２と同様に、論理セクタのロックを行う。

【０１２５】次に、ステップＳ３０３で、記憶場所管理テーブルを検索して、Ｎセクタに有効なデータが記録されているかどうかを判断する。有効なデータが記録されていればステップＳ３０４へ、記録されてないならばステップＳ３０５へそれぞれ分岐する。ステップＳ３０４では、それまで有効なデータとして記録されていたフラッシュＲＯＭ（または主記憶）のデータを破棄する。ステップＳ３０４におけるデータ破棄の処理は、図２１のフローチャートを用いて詳しく説明を加える。ステップＳ３０４の後ステップＳ３０５へ制御が移る。

【０１２６】ステップＳ３０５では、フラッシュＲＯＭ１５においてＮセクタを書き込むための記憶領域を獲得する。ステップＳ３０５における記憶領域の獲得手順は図２３を用いて詳しく説明を加える。ステップＳ３０５で正常に記憶領域の獲得に成功すれば、ステップＳ３０８へ制御を移す。ステップＳ３０８では獲得したフラッシュＲＯＭ１５の領域へＮセクタのデータを書き込む。

【0127】一方、ステップS305でフラッシュROMに記憶場所が無い場合、即ち記憶領域の獲得に失敗した場合はステップS306へ分岐する。ステップS306はデータの退避用に主記憶を獲得する。主記憶の領域確保はオペレーティングシステムが提供するメモリ管理機能によって行う。これはC言語でａｌｌｏｃ関数に相当する機能である。そして確保した領域を片方向線形リスト構造によって管理する。

【0128】図20は主記憶上に獲得した退避データリストの様子である。（a）は退避データリストにデータが無い状態であり、リストには、END_OF_LIST が代入されている。（b）は退避データリストに、セクタ番号3，20，221の各セクタの内容が退避されている状態である。

【0129】ステップS309で、記録場所管理テーブルを更新する。ここで、記録したフラッシュROM（または主記憶）へのポインタが代入される。ステップS310で論理セクタのアンロックを行う。図中点線で囲まれた期間その論理セクタを占有できる。ステップS311で記憶効率の評価を行う。記憶効率の評価手順については、図21のフローチャートを用いて詳しく説明を加える。記憶効率の評価の結果、記憶効率が悪化した場合は、ステップS312へ制御を移す。ステップS312では上述したガベージコレクションを行う。ガベージコレクションについては、図24のフローチャートを参照して詳しく説明を加える。ステップS313でNセクタの書き込みが終了してメインのルーチンへ復帰する。

【0130】なお、記憶場所管理テーブルに納められるのは、記憶場所のポインタ（バス空間上のアドレス）である。図20の（b）の主記憶に待避されたデータの「次のデータへのポインタ」の次のフィールド（図中ではすぐ下に示されている）からは、図10の左側にあるフラッシュROM上のデータ構造と互換性がある。記憶場所管理テーブルに納められるのはこの互換部分へのポインタである。このように構成することにより、データの読み出しプログラム側でフラッシュROMと主記憶を単一のアルゴリズムで扱うことが可能となる。

【0131】次に、指定されたセクタの記憶を破棄する手順（上述のステップS304）を説明する。図21は、記憶を破棄する手順を表わすフローチャートである。

【0132】ステップS401で指定領域の記憶破棄を開始する。ステップS402では、指定されたセクタを記憶する領域が主記憶上にあるかどうかを判断する。主記憶上にあるならステップS405へ分岐する。ステップS405で待避セクタリスト（本例では、図20で示した片方向線形リスト）から指定領域を削除する。

【0133】片方向線形リストからの指定領域の削除手順は、まずリストの先頭から検索方向順にリストをたどり、ポインタが自分をポインティングしている項を検出する。そして、この検出された項のポインタに現在自分がポインティングとしている値を代入することで実現する。そして、ステップS406で、リストから削除した主記憶領域をオペレーティングシステムへ返却する。オペレーティングシステムへの記憶領域の返却はC言語のｆｒｅｅ関数に相当する機能である。

【0134】一方、ステップS402で指定された領域が主記憶上でない（すなわちフラッシュROM上）ならステップS403へ分岐する。ステップS403では、指定されたフラッシュROM上のセクタの管理フラグを“使用済”へ変更する。これは、使用済みフラグをTRUEにセットすることで達成される。ステップS404では主記憶上の未使用セクタカウンタの値を1つ減少させる。ステップS407で復帰する。

【0135】次に、記憶効率の評価手順（ステップS311）について説明する。図22は、記憶効率の評価手順を表わすフローチャートである。

【0136】ステップS501で記憶効率の評価を開始する。ステップS502では、主記憶に設定された未使用セクタカウンタの値と使用済セクタカウンタの値を比較する。ここで、使用済セクタカウンタの値が未使用セクタカウンタの値に対して同じか上回った場合、上位プログラムに対して記憶効率の悪化をレポートする様に構成している（ステップS502、S504）。また、未使用セクタカウンタの値が使用済セクタカウンタの値よりも大きければ、評価結果を正常とし、正常復帰する（ステップS503）。

【0137】次に、フラッシュROMの記憶領域の獲得手順（ステップS305）について説明する。図23はフラッシュROMの記憶領域の獲得手順を表わすフローチャートである。

【0138】ステップS601でフラッシュROMの記憶領域の獲得を開始する。ステップS602で未使用セクタの検索権利を獲得する。ここでは、オペレーティングシステムの提供するセマフォの機能を使用して未使用セクタの検索権利を管理している。ここでは、ステップS602からステップS609／ステップS611までの点線で囲まれた処理期間だけ未使用セクタの検索権利を独占出来る。複数のタスクが同時に同一領域を獲得する様な事態を防ぐ為のしくみである。

【0139】ステップS603でフラッシュROMの最初のセクタへポインタを移動する。ステップS603でそのセクタの管理フラグ（使用中フラグ、使用済みフラグ）を参照して、当該セクタの使用状態を判断する。使用済みか使用中ならステップS605へ分岐する。ステップS605で現在ポイントしているセクタが最後のセクタならステップS611へ分岐する。この場合、使用可能な領域がフラッシュROM15に存在しないことになるので、ステップS611で未使用セクタの検索権利を開放した後、ステップS612で異常復帰する。ま

た、ステップＳ６０５で現在ポイントしているセクタが最後のセクタでなければ、ステップＳ６０６へ分岐する。ステップＳ６０６では、ポインタを次のセクタへ移動させてからステップＳ６０４へ戻る。

【０１４０】ステップＳ６０４ポインタの示すセクタの管理フラグが未使用となっていればステップＳ６０７へ分岐する。ステップＳ６０７では、フラッシュＲＯＭの管理フラグを“使用中”へ変更する（使用中フラグをＴＲＵＥにする）。そして、ステップＳ６０８で、主記憶に設けた未使用セクタカウンタの値を１つ減少させる。この場合は、フラッシュＲＯＭへの記憶領域の獲得に成功しているので、ステップＳ６０９で未使用セクタの検索権利を開放し、ステップＳ６１０で正常復帰する。

【０１４１】次に、ガベージコレクション（ステップＳ３１２）の手順について説明する。図２４はガベージコレクションの手順を表わすフローチャートである。

【０１４２】ステップＳ７０１でガベージコレクションを開始する。ステップＳ７０２では、整理対象のイレースブロック（以後、整理対象ブロック）を選出する。整理対象ブロックの選出手順については、図２５のフローチャートを用いて詳しく説明を加える。ステップＳ７０３では、整理対象ブロックの未使用セクタを使用済化する。この使用済化の手順については、図２６のフローチャートを用いて詳しく説明を加える。ここで、最初に整理対象ブロック内の未使用セクタを使用済化させる目的は、ガベージコレクション中であっても、他のタスクが整理対象ブロック内のセクタを含むセクタへの読み書きが可能な構成となっており、ガベージコレクション中に他のタスクによって整理対象ブロック内のセクタへ新たなデータが書き込まれることを防止する為である。

【０１４３】ステップＳ７０４では、整理対象ブロック中の使用中セクタを他の記憶領域（即ち、他のイレースブロック）へ移動させる。使用中セクタを他の記憶領域へ移動させる処理については、図２７のフローチャートを参照して詳しく説明を加える。

【０１４４】続くステップＳ７０５では、使用中セクタの移動を終了した整理対象ブロックの消去を実行する。整理対象ブロックを消去する手順については、図２８のフローチャートを参照して詳しく説明を加える。なお、この整理対象ブロックの消去において、消去回数カウンタ１５２の内容を主記憶にコピーしておく。ステップＳ７０５における整理対象ブロックの消去を終えると、ステップＳ７０６で主記憶に退避したデータをフラッシュＲＯＭの当該イレースブロックの消去回数カウンタへ戻す。そして、ステップＳ７０７でガベージコレクションから復帰する。

【０１４５】次に、ガベージコレクションにおける整理対象ブロックの選出手順（ステップＳ７０２）について説明する。図２５は整理対象ブロック選出する手順を表わすフローチャートである。

【０１４６】まず、ステップＳ８０１で整理対象ブロックの選出を開始する。ステップＳ８０２で評価ポインタに最初のイレースブロックをセットする。同様に、ステップＳ８０３で、整理対象候補ポインタを最初のイレースブロックにセットする。

【０１４７】次に、ステップＳ８０４で、評価ポインタの示すイレースブロックに使用済セクタが含まれているかどうかを判断する。使用済セクタが含まれていなければステップＳ８０４，ステップＳ８０５をスキップしてステップＳ８０７へ制御を移す。

【０１４８】一方、ステップＳ８０４で評価ポインタの示すイレースブロックに使用済セクタが含まれている場合には、ステップＳ８０５へ制御を移す。ステップＳ８０５では、整理対象候補ポインタの示すイレースブロックの消去回数カウンタの値と評価ポインタの示すイレースブロックの消去回数カウンタの値を比較する。もし評価ポインタの示すイレースブロックの消去回数の方が少なければステップＳ８０６へ制御を移す。ステップＳ８０６では、整理対象候補ポインタへ評価ポインタを代入する。一方、ステップＳ８０５でもし評価ポインタの示すイレースブロックの消去回数のほうが多ければそのままステップＳ８０７へ制御を移す。

【０１４９】ステップＳ８０７で評価ポインタが最後のイレースブロックを示しているかどうかを判断する。もし最後のイレースブロックでなければ、ステップＳ８０８で評価ポインタを次のイレースブロックへ移動させた後、ステップＳ８０４へ戻る。以上のように、ステップＳ８０４～Ｓ８０８の処理を繰り返すことで、整理対象候補ポインタは、使用済みセクタを含み、消去回数の少ないイレースブロックを示すようになる。

【０１５０】ステップＳ８０７で評価ポインタが最後のイレースブロックを示している場合はステップＳ８０９へ分岐する。ステップＳ８０９ではガベージコレクション処理（図２４の処理）に復帰する。この時点の整理対象候補ポインタの示すイレースブロックが整理対象として選出される。

【０１５１】次に、選択された整理対象ブロック内の未使用セクタを使用済み化する処理（ステップＳ７０３）について説明する。図２６は、整理対象ブロックの未使用セクタを使用済み化する手順を表わすフローチャートである。

【０１５２】ステップＳ９０１で処理を開始する。ステップＳ９０２で、整理対象ブロックの最初のセクタへポインタを移動させる。次に、ステップＳ９０３で、未使用セクタの検索権利を獲得する。これは、図２３のフローチャートのステップＳ６０２と同様の効果があり、ステップＳ９０８までの点線で囲まれた間、未使用セクタの検索権利を独占する。即ち、整理対象ブロックの全セクタを対象にスキャンして未使用セクタを使用済セクタへ変更するまでの間、他のタスクが未使用セクタの検索

をすることを禁止する。しかし、管理フラグのみの操作で未使用セクタを使用済セクタへ変更するので、検索権利の独占時間は短く、全体のスループットが低下することはない。

【０１５３】ステップＳ９０４で、現在のポインタが示すセクタが未使用セクタかどうかを判断する。もし未使用セクタならステップＳ９０５へ分岐する。ステップＳ９０５でその記憶を廃棄する。ステップＳ９０５の処理手順は図２１のフローチャートで説明した通りである。この処理により、未使用セクタが使用済みセクタに変更される。ステップＳ９０６では、ポインタが整理対象ブロックの最後のセクタを示しているかどうかを判断する。最後のセクタを示していればステップＳ９０８へ、そうでないならステップＳ９０７へ分岐する。ステップＳ９０７ではポインタを次のセクタへ移動させてステップＳ９０４へ制御を戻す。

【０１５４】また、ステップＳ９０６でポインタが整理対象ブロック最後のセクタならステップＳ９０８で未使用セクタの検索権利を開放し、ステップＳ９０９でガベージコレクション処理（図２４のフローチャート）へ復帰する。

【０１５５】次に、整理対象ブロックの使用中セクタを他のイレースブロックの未使用セクタへ移動する処理（ステップＳ７０４）について説明する。図２７は、整理対象ブロックの使用中セクタの移動手順を表わすフローチャートである。

【０１５６】ステップＳ１０００で処理を開始する。ステップＳ１００１で整理対象ブロックの最初のセクタへポインタを移動させる。以下のステップＳ１００２～Ｓ１０１２では、ポインタが指し示すセクタについて処理を行う。

【０１５７】ステップＳ１００２で当該セクタの管理フラグ（使用中フラグ、使用済みフラグ）を判断する。ステップＳ１００２で管理フラグの値が「使用中」となっていたらステップＳ１００３へ制御を移し、「使用済」となっていたらステップＳ１０１２へ制御を移す。ステップＳ１００３で、論理セクタをロックする。ロックしたセクタはステップＳ１０１１でアンロックされるまでの間、自タスクで占有される。

【０１５８】ステップＳ１００４では記憶領域を獲得する。ステップＳ１００４における記憶領域の確保の手順は、図２３のフローチャートで説明した通りである。ここで、整理対象ブロック内の各セクタは上記ステップＳ７０３の処理で、全て使用済み化されているので、確保される記憶領域は整理対象ブロック以外のイレースブロックとなる。

【０１５９】記憶領域の獲得に成功すると、処理はステップＳ１００８へ進む。ステップＳ１００８では、獲得した領域へ当該セクタのデータをコピーする。そして、セクタの移動に従って、ステップＳ１００９で記憶場所管理テーブル１４０を更新する。

【０１６０】一方、ステップＳ１００４で記憶領域の獲得に失敗した場合は、ステップＳ１００５へ分岐する。ステップＳ１００５では、データ退避用の記憶領域を主記憶（ＤＲＡＭ）より獲得する。データ退避用記憶領域の獲得は図１９のフローチャートのステップＳ３０６で説明した通りである。ステップＳ１００６では、獲得した領域へ当該セクタのデータをコピーする。そして、ステップＳ１００７で記憶管理テーブルを更新する。ステップＳ１０１０で元の記憶を廃棄する。即ち、ポインタの指し示すセクタの使用済みフラグをＴＲＵＥにセットする。そして、ステップＳ１０１１で当該論理セクタをアンロックする。

【０１６１】ステップＳ１０１２で、ポインタの指し示すセクタが、整理対象ブロックの最後のセクタかどうかを判断する。最後のセクタであればステップＳ１０１４へ、最後のセクタでなければステップＳ１０１３へそれぞれ分岐する。ステップＳ１０１３では、ポインタを次のセクタへ移動させて、ステップＳ１００２へ戻り、次のセクタについて上述の処理を繰り返す。また、ステップＳ１０１４では、整理対象ブロック内の全てのセクタについて処理を終えているので、ガベージコレクション処理（図２４のフローチャート）へ復帰する。

【０１６２】次に、整理対象ブロックの消去処理（ステップＳ７０５）について説明する。図２８は、整理対象となったイレースブロックの消去手順を表わすフローチャートである。

【０１６３】ステップＳ１１０１で処理を開始する。ステップＳ１１０２で整理対象ブロックの消去回数カウンタを主記憶へコピーする。ステップＳ１１０３では整理対象ブロックの消去を実行する。ステップＳ１１０４では、主記憶へコピーした消去回数カウンタの値を１増加させた値をフラッシュＲＯＭへ書き込む。即ち、当該整理対象ブロックの消去カウンタの値を、消去処理前の値より１増加させる。その後、ステップＳ１１０５でガベージコレクション処理（図２４のフローチャート）へ復帰する。

【０１６４】上記図２４で示されるガベージコレクション処理は、極力フラッシュＲＯＭを用いた処理であり、退避データの安全性が高い。しかしながら、上述の<未使用セクタが無い場合>の項で説明したように、積極的に主記憶（ＤＲＡＭ１４）を用いて使用中セクタのデータを待避し、複数個のイレースブロックを消去すると消去処理の効率がよい。但し、ＤＲＡＭ１４にデータを待避するので、待避中のデータに関して安全性が低下する（例えば電池が外れて電源供給が停止するとＤＲＡＭに待避したデータが失われることになる）。そこで、電源の種別を判断し、供給電源が電池の場合は待避データの安全性を重視し、ＡＣアダプタの場合は電源供給が停止する危険性が少ないので消去処理の効率を重視するよう

に構成してもよい。この場合の処理について図３６を参照して説明する。

【０１６５】図３６は、電源種別に基づいてガベージコレクション処理を切り換える場合の処理手順を説明するフローチャートである。同図において、図２４のフローチャートで示される処理と同じ処理を行うステップについては同一のステップ番号を付し、ここでは詳細な説明を省略する。

【０１６６】ステップＳ１３００においてガベージコレクション処理が起動されると、ステップＳ１３０１へ進み、当該装置への電源供給の形態を判断する。ここでは、図１の電源コントローラ９が、電源の供給元が電池７であるかＡＣアダプタ２３であるかを判断し、ＣＰＵ５に通知する。電源種別が電池７であった場合は、ステップＳ１３０４へ進み、上述の図２４で示したガベージコレクション処理を実行する。

【０１６７】一方、ステップＳ１３０１において電源種別がＡＣアダプタであった場合は、ステップＳ１３０２へ進む。ステップＳ１３０２では、図２４のステップＳ７０２、Ｓ７０３、Ｓ７０４に相当する処理を実行し、選出した整理対象ブロック内の未使用セクタの使用済み化と使用中セクタの待避を行う。そして、ステップＳ１３０３において、主記憶（ＤＲＡＭ１４）にセクタの待避を行うのに十分な空き領域があるか否かを判断し、十分な空き領域があればステップＳ１３０２へ戻る。ステップＳ１３０２では、前回の整理対象ブロックとは別の整理対象ブロックを選出して、上述の処理を繰り返す。

【０１６８】ＤＲＡＭ１４上に十分な空き領域が無くなると、ステップＳ１３０３からステップＳ１３０４へ進み、上述の処理で選出された整理対象ブロックの消去を行う。そして、ステップＳ７０６で主記憶に待避したデータをフラッシュＲＯＭ１５に戻して本処理を終了する。

【０１６９】以上のように、図３６の処理によれば、電源がＡＣアダプタによって供給される場合は、主記憶の空き容量を積極的に利用してデータの待避を行い、複数の整理対象ブロックを選出して、一括して消去処理を行うことができ、消去処理の効率が向上する。

【０１７０】なお、上記の処理では、電源種別に基づいて自動的にガベージコレクションの形態を切り換えるが、コントロールパネル１２の操作により、マニュアルで切り換えるようにすることもできることはいうまでもない。

【０１７１】次に、基本サービスの一つである論理セクタの解放手順について説明する。図２９は、論理セクタの解放手順を表すフローチャートである。

【０１７２】ステップＳ１２０１でＮセクタの解放を開始する。ステップＳ１２０２でＮセクタをロックする。この結果、ステップＳ１２０５でアンロックされるまでの間、自タスクで論理セクタを占有出来る。続いて、ステップＳ１２０３で当該セクタの記憶を廃棄する。この記憶の廃棄処理については、図２１のフローチャートで説明した通りである。ステップＳ１２０４では、ＤＲＡＭ１４の記憶場所管理テーブル１４０へ“不在”値を代入する。ステップＳ１２０５では、論理セクタをアンロックし、ステップＳ１２０６で復帰する。

【０１７３】例えばＭＳ－ＤＯＳ（商標）等の一般のファイルシステムでは、ファイルの消去に際しては、当該ファイルに属するセクタをＦＡＴにおいて上書き可能とするのみで、各セクタを解放するということは行われない。よって、このようなファイルシステムに本実施形態のフラッシュＲＯＭ管理システムを適用すると、ファイルシステム上では無効となったデータが、有効なセクタとして残されてしまうことになり、ガベージコレクション等の効率を低下させることになる。よって、ファイルシステムの指示（例えばファイル消去）に基づいて不要となったセクタを検出し、これを解放するように構成すれば、ガベージコレクションの効率をより向上させることができる。

【０１７４】図３７は、ファイルシステムよりファイル消去が指示された場合の、不要セクタの解放手順を表すフローチャートである。同図において、ステップＳ１４０１でファイルシステムよりファイル消去の指示があったか否かを判断する。ファイル消去の指示があった場合は、ステップＳ１４０２へ進み、消去すべく指示されたファイルに含まれるセクタを抽出する。セクタの抽出は、ＦＡＴを参照することで抽出できる。そして、ステップＳ１４０３で、先のステップＳ１４０２で抽出された各セクタについて、上記図２９のフローチャートで説明したセクタの解放処理を実行する。

【０１７５】［実施形態２］次に実施形態２について説明する。

【０１７６】＜ディスクコントローラエミュレーション＞上述の実施形態１で説明したフラッシュＲＯＭの記憶管理システムは、上位層から見た特徴がディスク媒体と良く似ている。従って、ディスクコントローラのエミュレーション機能を備えたシステムに組み込むことで、ディスクコントローラとディスク媒体をディスクコントローラエミュレーションと本実施形態の記憶管理システム（あるいは本実施形態の記憶管理システムを組み込んだＩＣカード）へ置き換えることが可能となる。近年ＰＣＭＣＩＡに代表されるＩＣカードが普及しているが、ＩＣカードへディスクコントローラエミュレーション機能と上記実施形態１の記憶管理システムを組込むことにより、リムーバルな記憶媒体として利用することが可能となる。第２の実施形態では、実施形態１の記憶管理システムをＩＣカードへ組み込んだものについて説明する。

【０１７７】図３０は実施形態２におけるＩＣカードの構成を表すブロック図である。同図において、２００はＩＣカード全体を示す。２０１はマイクロコンピュータ

であり、ディスクコントローラエミュレーション及び記憶管理を行う。２０２はＲＯＭであり、マイクロコンピュータ２０１のプログラムを格納する。２０３はＲＡＭであり、マイクロコンピュータ２０１の主記憶として機能する。２０４はフラッシュＲＯＭであり、上記実施形態１で説明した記憶管理システムによってデータを蓄積する。即ち、フラッシュＲＯＭ２０４は、図４で説明した管理領域とデータ領域とで管理される。

【０１７８】２０５はコマンド／データ・ラッチ部であり、ホスト装置より受信した外部バスからのコマンドとシリンダ番号等を保持する。２０６はＦＩＦＯメモリであり、先入れ先出し方式でデータの入出力を行う。２０７はタプルＲＯＭであり、当該カードの特徴等を記憶しており、外部バスからのみ読み出しができる。

【０１７９】上述の各構成の機能は、以降の動作説明でより明らかとなる。

【０１８０】図３１は、本実施形態２のＩＣカードを利用する為のホストシステムの簡単なブロック図である。同図において、３０１はホストシステム側のマイクロコンピュータである。３０２はカードインターフェースであり、ホストシステムの内部バスとＩＣカード２００の外部バスを接続する。なお、カードインターフェース３０２は、ＩＣカード２００への電源供給を行うための電源供給線や、ＩＣカード２００からの割り込み要求（ＩＲＱ出力）を受け付けるための信号線も備えている。

【０１８１】図３２は、図３１のホストシステムがＩＣカードを接続する際の手順を示すフローチャートである。ステップＳ４１００で処理を開始すると、ステップＳ４１０１でＩＣカードへの電源供給を開始する。ステップＳ４１０２では、ＩＣカード２００内のタプルＲＯＭ７から、タプル形式で格納されているデータを解析する。タプルＲＯＭ７の内容を解析することで、接続されているＩＣカードの特徴が分かる。

【０１８２】ステップＳ４１０３では、ステップＳ４１０２で解析したタプル情報によって、接続されているＩＣカードが内部バスへ接続可能かどうかを判断する。そして、接続可能ならステップＳ４１０４へ、接続不可能ならステップＳ４１０５へとそれぞれ分岐する。ステップＳ４１０４では、ＩＣカード側のバスをホストの内蔵バスのメモリ空間とＩＯ空間へマッピングする。この時点でホスト装置のバスの空間にディスクコントローラが有るのと同じ状態になる。

【０１８３】図３３はＩＣカード２００内のマイクロコンピュータ１のメインシーケンスを示すフローチャートである。ステップＳ４２０１でＩＣカードの電源が投入されると、ステップＳ４２０２で記憶管理システムの初期化を行う。即ち、フラッシュＲＯＭ２０４の全イレースブロックの論理セクタの状態を一旦読み出し、読み出した情報に従って主記憶用のＲＡＭ２０３へ記憶場所管理テーブルを作成する。ステップＳ４２０３で主記憶上のコマンドバッファとしてリング状のバッファを用意して初期化し、割り込み処理を許可する。この処理以降割り込みルーチンの動作が始まる。

【０１８４】割り込みルーチンのシーケンスを図３４のフローチャートに示す。割り込みルーチンの動作を理解した方が、図３３のフローチャートの説明が容易となる為、ここで図３４のフローチャートについて説明を行う。

【０１８５】ホストシステムがコマンド／データ・ラッチ２０５へのコマンドのアドレスへコマンドを書き込むと、コマンド／データ・ラッチ２０５からマイクロコンピュータ２０１へ割り込みが発生する。コマンド／データ・ラッチ２０５は、ホストバスとＩＣカード内部のバスのＩＯアドレス空間にマッピングされていて、コマンド／データはそれぞれ図３５に示すようにＩＯアドレスが割り振られている。図３５は、本実施形態のコマンド／データ・ラッチにおけるＩＯ割り付けを示す図である。本例では、図３５中のCommandのアドレスにコマンド（例えばデータの読出しを指示するReadSector(s)）を書き込むことでマイクロコンピュータ２０１へ割り込みが発生する。

【０１８６】割り込みが発生すると、マイクロコンピュータ２０１のソフトウェアは、図３４のフローチャートのステップＳ４３０１へ制御を移す。ステップＳ４３０２では、コマンド／データ・ラッチ２０５に書き込まれたデータを読み出して、主記憶上のリングバッファへデータを格納する。ステップＳ４３０３で割り込みルーチンを終了して図３３のフローチャートへ復帰する。

【０１８７】図３３のフローチャートの説明に戻る。ステップＳ４２０４でマイクロコンピュータ２０１はコマンドバッファの状態を判断する。コマンドバッファへデータが格納されていれば、ステップＳ４２０５へ分岐し、データが格納されていなければステップＳ４２１３へ分岐する。ステップＳ４２１３ではＣＰＵを休止状態にする。多くのワンチップマイクロコンピュータは、命令の実行を休止して消費電流を減らす機能を備えているが、本実施形態のＣＰＵもこの種の機能を備える。そして、ＩＲＱによる割り込み要求信号が入力されると、ＣＰＵ２０１は休止状態から復帰して上述の割り込みルーチンを実行する。割り込みプログラムの実行が済んだ時点でステップＳ４２１３から復帰してステップＳ４２０４へ戻る。

【０１８８】ステップＳ４２０４でコマンドバッファへデータが格納されていると、ステップＳ４２０５へ移行する。ステップＳ４２０６では、リングバッファからデータを読み出す。ステップＳ４２０６でコマンドを解釈する。Seekコマンドの場合はステップＳ４２０７、ReadSector(s)コマンドの場合はステップＳ４２０８へ、WriteSector(s)の場合はステップＳ４２０９へ、IdentifyDrvコマンドの場合はステップＳ４２１０へそれぞれ分岐

する。他にもコマンドがあるが本実施形態の説明上重要でないものは省き、フローチャートを簡略化している。ステップＳ４２０７～４２１０までのコマンドの実行を終了したらステップＳ４２０４まで戻り、上記の処理を繰り返す。

【０１８９】ステップＳ４２０７では、Seekコマンドを実行する。Seekといってもフラッシュ ＲＯＭには、ディスクデバイスと違ってヘッドが無いので、次のコマンドに備えての妥当性等をチェックするだけである。ＩＣカードのサポートするヘッド数を超えるヘッド位置などを指定された場合は、ディスク装置同様にエラーが発生する。

【０１９０】ステップＳ４２０８はReadSector(s)コマンドに対する処理を行う。ReadSector(s)コマンドは、読み出すべきセクターの個数が図３５のSectorCountで指定される。よって、ステップＳ４２０８では、指定された場所のセクタをSectroCount個読み出す行為を行う。本実施形態の記憶管理システムでは、リニアな論理セクタ番号を使って管理を行っているので、シリンダ／ヘッド／セクタ番号を元にリニアな論理セクタ番号を計算し、論理セクタの内容をＦＩＦＯメモリ２０６へ転送し、コマンド／データ・ラッチ２０５のSectorNumberのインクリメントも行う。ＦＩＦＯメモリ２０６は、ＩＣカード２００の内部バスから書き込んだデータを外部バスから読み出すことができ、また外部バスから書き込んだデータをＩＣカード内部バスから読み出す構成となったＦＩＦＯメモリである。

【０１９１】ここで、上述のリニアな論理セクタ番号について説明する。一般にハードディスクに対して指定する番号は、セクタ、シリンダ、ヘッドのパラメータで決まる３次元の不連続な番号である。例えば、シリンダ数が１０２４個、ヘッド数が１６個、セクタ数が６３個のハードディスクの場合、セクタ数は１０２４×１６×６３＝１０３２１９２個となる。

【０１９２】このセクタを０番から１０３２１９２番としてアクセスできると良いのであるが、上記の３つのパラメータをすべて指定してアクセスするように設計されている。例えば、シリンダ５００・ヘッド１６・セクタ６３の次は、シリンダ５０１・ヘッド０・セクタ１をアクセスするといった具合である。なお、これら３つのパラメータをそれぞれの頭文字をとってＣＨＳパラメータと呼ぶ。

【０１９３】ＭＳ－ＤＯＳ（商標）のようなオペレーティングシステムでは、内部ではリニア（連続的）なセクタ番号を用いるが、デバイスドライバがこれをＣＨＳパラメータに変換する。本実施形態のシステムでは、リニアなセクタ番号を用いるのでＣＨＳパラメータの値を元にリニアなセクタ番号を求める。上記で挙げたハードディスクの場合は、

シリンダ番号×（１６×６３）＋ヘッド番号×（６３）＋セクタ番号

を計算することで、リニアな論理セクタ番号が求まる。

【０１９４】ステップＳ４２０９はデータラッチで指定された場所のセクタへデータを書き込む処理を行う。データは、ＦＩＦＯメモリ２０６経由でホストシステムから受け取る。

【０１９５】ステップＳ４２１０は、ＩＣカード２００がどのようなハードディスクをエミュレーションしているかという情報を返す処理を行う。すなわちシリンダ数やModelNumberなどハードディスクとしてのスペックを含むデータをＦＩＦＯメモリ２０６へ書き込む処理を行う。

【０１９６】＜ファイルシステムの解析＞以上説明した様に実施形態１で説明した記憶管理システムをＩＣカードに組み込むことで、ＡＴＡハードディスク等の置き換え用途に使用できる。しかし、ＡＴＡコマンド等のＦＡＴキャッシュやファイル消去によって生じた不要セクタの開放といった処理を行う為の情報を上位システムからもらう手法が無い。ＡＴＡコマンドの空き部分を利用してセクタの開放コマンドとキャッシュするセクタ番号指定コマンドを追加実装することで、ＦＡＴキャッシュと不要セクタ解放の機能が実現できる。そして、この様な機能があることを想定していない現状のＭＳ－ＤＯＳ等のシステムでも、ＦＡＴキャッシュやセクタの開放を実現できた方が良いことは言うまでもない。

【０１９７】ＦＡＴシステムは論理セクタ番号０に相当する部分にＦＡＴの場所やサイズといった情報を格納している。本実施形態では、このセクタを読むことでＦＡＴの場所やサイズを取得し、ＦＡＴキャッシュの処理に利用する。同様に本来書き込みデータの内容を理解しないはずのＩＣカードであるが、ファイルシステムの為の情報（ディレクトリエントリやＦＡＴ）を解析することでＩＣカードが自立的に不要セクタを判断して開放する等の処理に役立てることが出来る。もちろんＦＡＴに限った話では無く、ＨＰＦＳやマッキントッシュ（商標）のファイルシステムでも書き込むデータの内容を解析すれば、不要セクタの検出が可能である。この様に構成することでＡＴＡハードディスクのインタフェースでも、ファイルシステムの動作に合わせた最適化処理を行うことを可能にする。

【０１９８】上記装置の機能もしくは方法の機能によって達成される本発明の目的は、前述の実施形態のプログラムを記憶させた記憶媒体によっても達成できる。例えば、パーソナルコンピュータに、その記憶媒体を装着し、その記憶媒体から読み出した以下に説明するようなフラッシュＲＯＭ管理プログラムを実行することにより、フラッシュＲＯＭをディスクシステムと同等に使用できるようになるとともに、書き換え耐久の分散化を図れる。このための本発明にかかるプログラムの構造的特徴は、図４９に示す通りである。

【0199】図49は本実施形態における記憶媒体に格納される制御プログラムの制御手順、及び本記憶媒体のメモリマップを示す図である。

【0200】図49（a）において、350は管理処理であり、データ領域、及びデータ領域に対応する管理領域とで構成される複数のセクタをフラッシュROMに形成し、各セクタにおいてデータ領域の記憶状態を示す状態情報（セクタ番号154、使用中フラグ155、使用済フラグ156）を管理領域に格納し、該状態情報に基づいてフラッシュROM15のアクセスを管理する。例えば、図18や図19のフローチャートで示したように、セクタ単位でのフラッシュROMへのデータの書込みや読出しを制御する。

【0201】351は消去処理であり、フラッシュROM内のイレースブロックについて、状態情報に基づいてデータ領域に有効データが存在するセクタを抽出し、その内容を移動させて当該イレースブロックを消去する。これは、図24のフローチャートで示したガベージコレクション処理である。

【0202】352はカウント処理であり、消去処理351による消去が行われた回数を消去回数としてイレースブロック毎にカウントし、該消去回数を各イレースブロックの消去回数カウンタ152に記憶する。これは、図28のステップS1102、S1104で示される処理である。

【0203】353は決定処理であり、消去処理351において消去対象となるべきイレースブロックを消去回数カウンタの値に基づいて決定する。これは、図25のフローチャートで示される処理である。

【0204】実際の制御手順は、管理処理350によってガベージコレクション処理の開始が指示されると、決定処理353が消去回数カウンタ152を参照して消去対象のイレースブロックを決定する。消去処理351で、決定処理で決定された消去対象のイレースブロックに対して消去処理を実行すると、カウント処理352によって、当該イレースブロックの消去回数カウンタ152が1つインクリメントされる。

【0205】上記制御手順を実現するための制御プログラムは、フロッピーディスクやハードディスク、あるいはCD－ROM等の記憶媒体に、例えば図49の（b）のメモリマップに示すような構成で格納される。上記制御プログラムは、例えばパーソナルコンピュータ等の情報処理装置によって読み出され、主記憶（RAM）上にロードされて、CPUにより実行される。なお、主記憶上への上記制御プログラムのロードは、LANを介して行われてもよい。

【0206】なお、図49（b）において、管理処理モジュール350'、決定処理モジュール353'、消去処理モジュール351'、カウント処理モジュール352'は、それぞれ制御手順で示した管理処理350、決定処理353、消去処理351、カウント処理352の各処理を実行するプログラムモジュールである。

【0207】また、本発明は、複数の機器から構成されるシステムに適用しても、1つの機器からなる装置に適用してもよい。また、本発明はシステム或は装置にプログラムを供給することによって達成される場合にも適用できることは言うまでもない。この場合、本発明に係るプログラムを格納した記憶媒体が、本発明を構成することになる。そして、該記憶媒体からそのプログラムをシステム或は装置に読み出すことによって、そのシステム或は装置が、予め定められた仕方で動作する。

【0208】

【発明の効果】以上説明したように、本発明によれば、フラッシュROMをファイルシステムに適応させることが可能な管理方式において、各消去単位毎の消去回数のばらつきを減少させることが可能となる。

【0209】

【図面の簡単な説明】

【図1】実施形態1におけるカメラシステムの構成を表すブロック図である。

【図2】本実施形態の電子カメラにおけるファイルシステムの階層構造を表す図である。

【図3】デバイスドライバの管理ブロックをC言語で記述した宣言文を示す図である。

【図4】フラッシュROM上のセクタ構造の例を示す図である。

【図5】管理領域用データと、データ領域とを分離して格納する構成を表す図である。

【図6】各フラグの状態に対応した意味を示す図である。

【図7】フラッシュROMにおけるセクタの書き換え手順を説明する図である。

【図8】本実施形態におけるフラッシュROMのガベージコレクション動作を説明する図である。

【図9】未使用セクタが存在しない場合のガベージコレクションの動作を説明する図である。

【図10】DRAM上に作成された記憶場所管理テーブルを説明する図である。

【図11】キャッシュソフトウエアの階層的な位置付けを表す図である。

【図12】キャッシュの主記憶上のデータ構造を表わす図である。

【図13】FATキャッシュの読み出し手順を表すフローチャートである。

【図14】FATキャッシュの書込み手順を表すフローチャートである。

【図15】データ書き込み完了を確認するための動作手順をC言語で表現した図である。

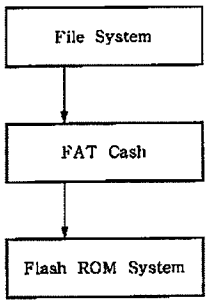
【図16】DC／DCコンバータの出力容量を越えない様に電源を管理するプログラムをC言語で表現した図で

ある。
【図１７】本実施形態のリブートからサービスの開始までの動作手順を表わすフローチャートである。
【図１８】指定セクタの読み出しサービスの手順を表わすフローチャートである。
【図１９】論理セクタの書き込みサービスの手順を表わすフローチャートである。
【図２０】主記憶上に獲得した退避データリストの様子である。
【図２１】記憶を破棄する手順を表わすフローチャートである。
【図２２】記憶効率の評価手順を表わすフローチャートである。
【図２３】フラッシュＲＯＭの記憶領域の獲得手順を表わすフローチャートである。
【図２４】ガベージコレクションの手順を表わすフローチャートである。
【図２５】整理対象ブロック選出する手順を表わすフローチャートである。
【図２６】整理対象ブロックの未使用セクタを使用済み化する手順を表わすフローチャートである。
【図２７】整理対象ブロックの使用中セクタの移動手順を表わすフローチャートである。
【図２８】整理対象となったイレースブロックの消去手順を表わすフローチャートである。
【図２９】論理セクタの解放手順を表すフローチャートである。
【図３０】実施形態２におけるＩＣカードの構成を表すブロック図である。
【図３１】本実施形態２のＩＣカードを利用する為のホストシステムの簡単なブロック図である。
【図３２】図３１のホストシステムがＩＣカードを接続する際の手順を示すフローチャートである。
【図３３】ＩＣカード内のマイクロコンピュータのメインシーケンスを示すフローチャートである。
【図３４】ＩＣカード内のマイクロコンピュータの割り＊込み処理の手順を表すフローチャートである。
【図３５】ＩＯアドレスの割り付け状態を表す図である。
【図３６】電源種別に基づいてガベージコレクション処理を切り換える場合の処理手順を説明するフローチャートである。
【図３７】ファイルシステムよりファイル消去が指示された場合の、不要セクタの解放手順を表すフローチャートである。
【図３８】本実施形態における消去処理速度向上のための前処理の制御手順を表すフローチャートである。
【図３９】本実施形態におけるフラッシュＲＯＭへの１バイトデータの書込み手順を表すフローチャートである。
【図４０】電源の共有手順を説明するためのフローチャートである。
【図４１】本実施形態におけるフラッシュＲＯＭへの制御プログラムの格納状態を説明する図である。
【図４２】相対アドレスで表現されたプログラムコードの一例を表す図である。
【図４３】図４２のリロケーション情報レコードのデータを格納するテーブルを表す図である。
【図４４】図４１のプログラムを主記憶の８７１０番地へマッピングした場合のプログラムコードを示す図である。
【図４５】ディレクトリスロットの特徴を表す図である。
【図４６】図４５のディレクトリスロットにおいて、FileB が削除された状態を示す図である。
【図４７】本実施形態のＤＯＳ互換ファイルシステムでファイルを消去した後の状態を表す図である。
【図４８】本実施形態による電源資源（セマフォ）の時分割利用を説明するフローチャートである。
【図４９】本実施形態の制御を実現する制御プログラムを提供する記憶媒体の内容を説明する図である。

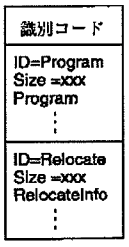
【図3】

```
1 : typedef  struct  DeviceTag  {
2 :        struct  DeviceTag  *Next :
3 :        char    DevName [32] :
4 :        int     (*InitDev) (void) :
5 :        int     (*ShutDown) (void) :
6 :        int     (*ReadSector) (long  lsect, long  nsect, char  *buffer) :
7 :        int     (*WriteSector) (long  lsect, long  nsect, char  *buffer) :
8 :        int     (*ReleaseSector) (long  lsect, long  nsect) :
9 : } Device :
```

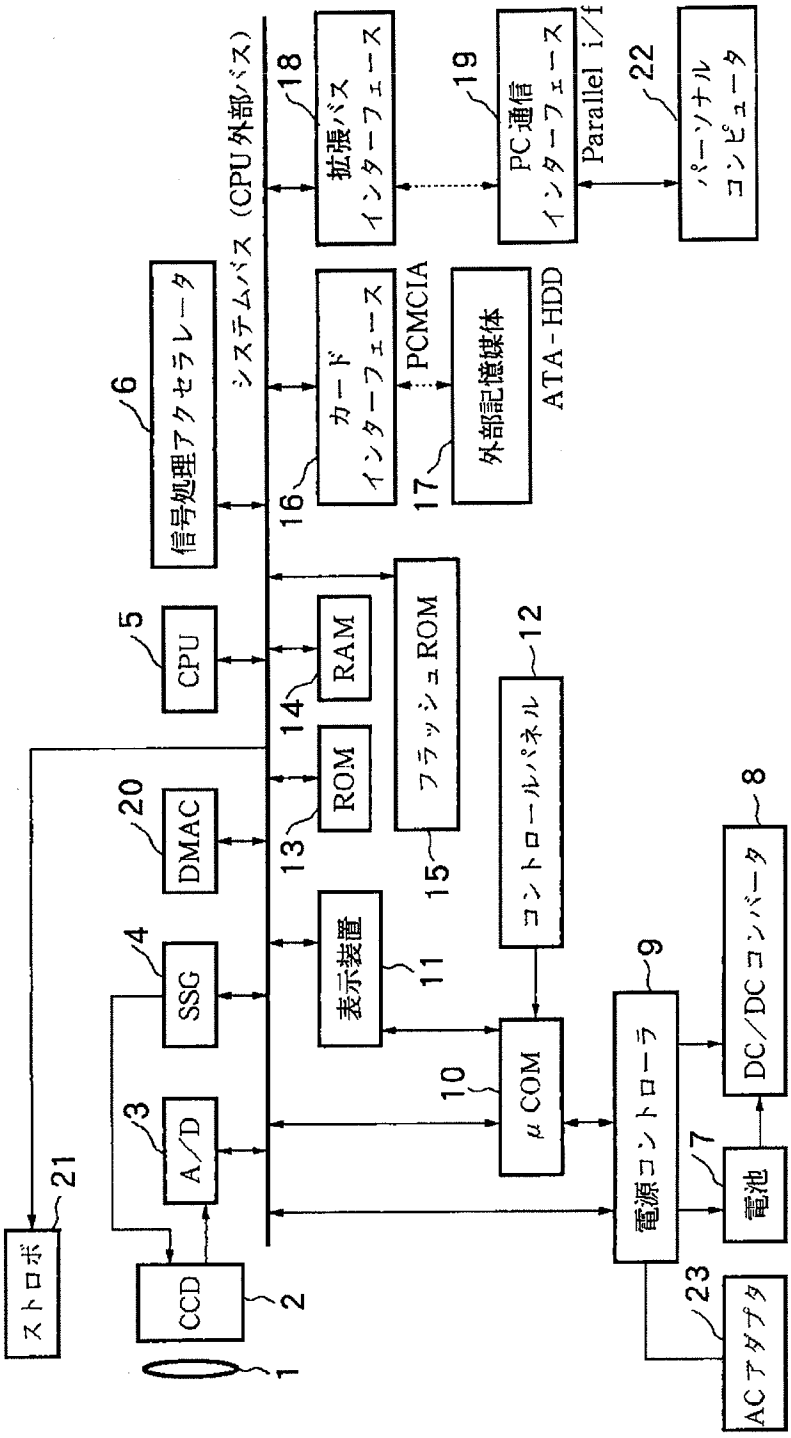
【図１１】



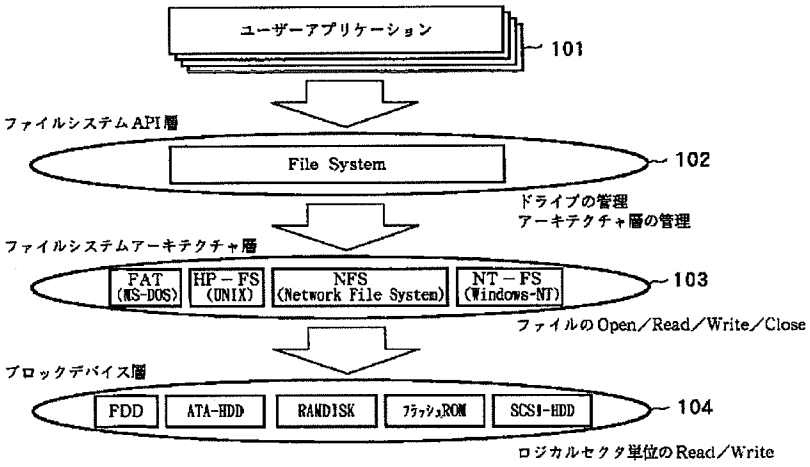
【図４１】



【図1】


ストロボ
21
CCD
2
1
A／D
3
SSG
4
DMAC
20
CPU
5
信号処理アクセラレータ
6
システムバス（CPU外部バス）
ROM
13
RAM
14
フラッシュROM
15
表示装置
11
μCOM
10
コントロールパネル
12
電源コントローラ
9
電池
7
DC／DCコンバータ
8
ACアダプタ
23
拡張バス
インターフェース
18
カード
インターフェース
16
PCMCIA
外部記憶媒体
17
ATA-HDD
PC通信
インターフェース
19
Parallel i／f
パーソナル
コンピュータ
22

【図2】

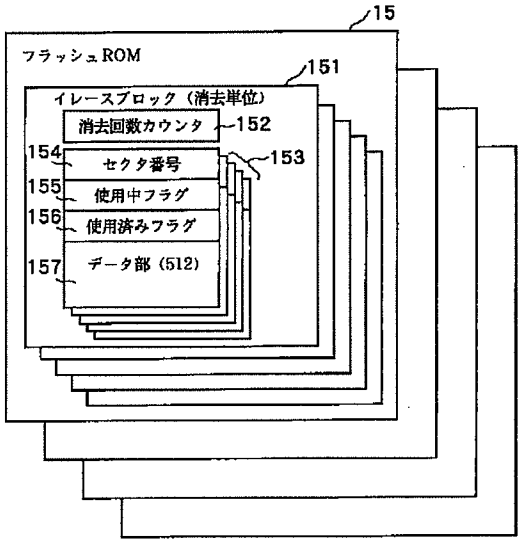


【図42】

相対アドレスで表現されたプログラムコード

番地	データ	ニーモニック
0040	1234	DB 1234
0050	85,0010	JMP 0010
⋮		
0060	86,-20	MOV A,@(-20)
⋮		

【図4】



【図15】

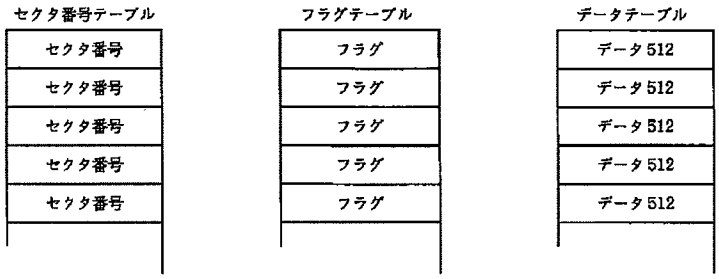
```
1 : void WriteByteEEP (struct DEV * Dev,char * Address,char Data)
2 : {
3 :       while (* Dev - > Address  ! = Dev - > Data) :
4 :       * Address = Data :
5 :       Dev - > Address = Address :
6 :       Dev - > Data     = Data :
7 :       RotateRdyQueue0 :
8 : }
9 : char ReadByteEEP (struct DEV * Dev,char * Address)
10 : {
11 :      if (Dev - > Address = = Address) return Dev - > Data :
12 :      while (* Dev - > Address  ! = Dev - > Data) :
13 :      return * Address :
14 : }
```

【図44】

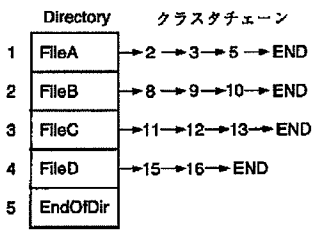
主記憶の8710番地へマッピングされたプログラムコード

番地	データ	ニーモニック
8750	1234	DB 1234
8760	85,8770	JMP 8770
⋮		
8770	86,8750	MOV A,@(8750)
⋮		

【図5】



【図45】

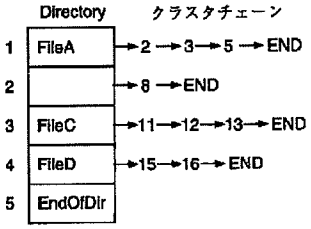


【図6】

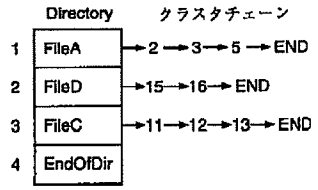
155 使用中フラグ	156 使用済みフラグ	意味
FALSE	FALSE	「未使用セクタ」...消去後の状態
TRUE	FALSE	「使用中セクタ」...セクタ番号が示すセクタをデータ部に格納している
TRUE	TRUE	「使用済みセクタ」...データ部に格納されているデータは無効であり消去するまでこのセクタを利用できない

FALSE：消去後の状態　TRUE：FALSEのビット反転値

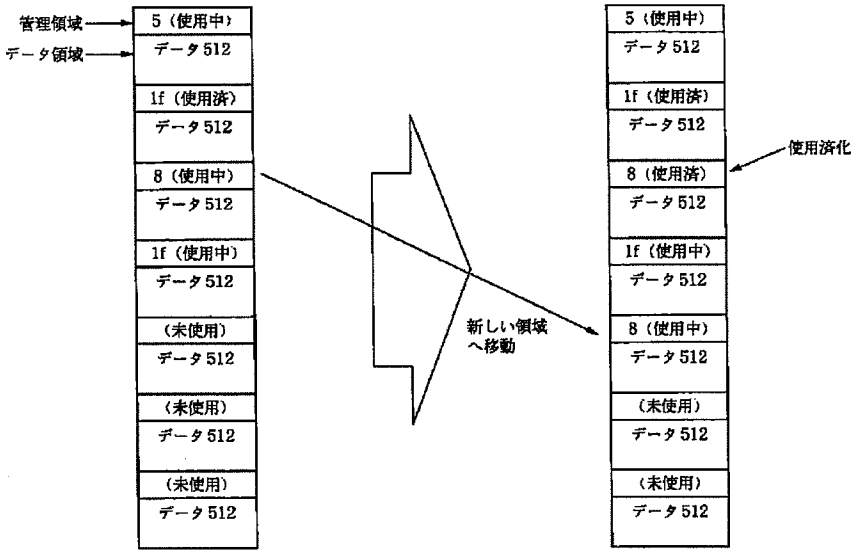
【図46】



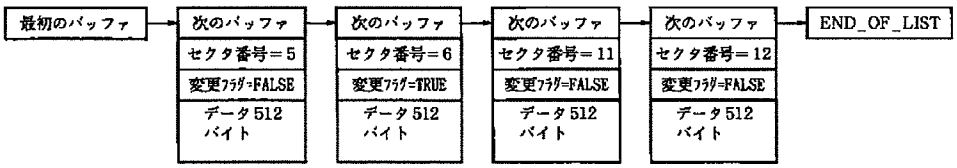
【図47】



【図7】



【図12】

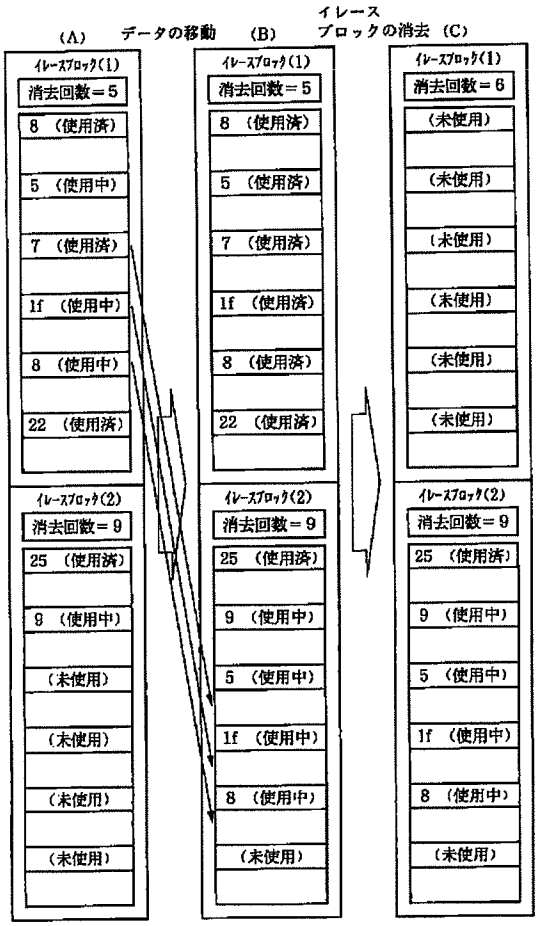


【図43】

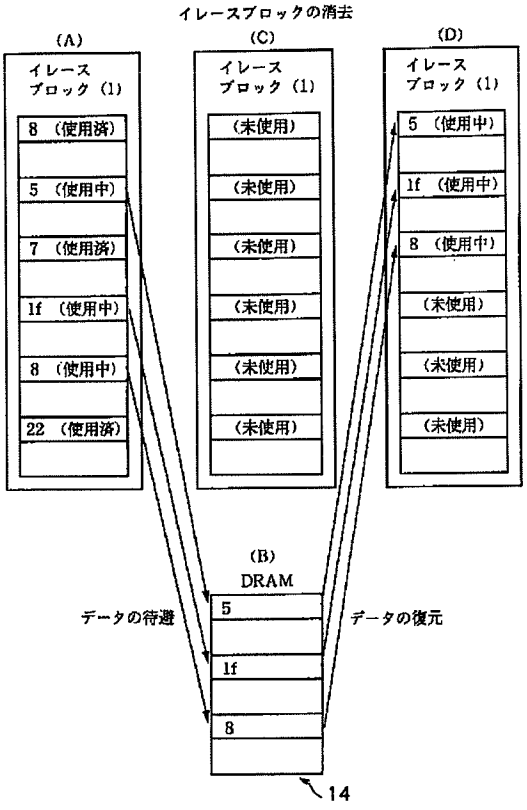
プログラム中で絶対アドレスに変換しなければならない番地のテーブル

0052
0062
⋮

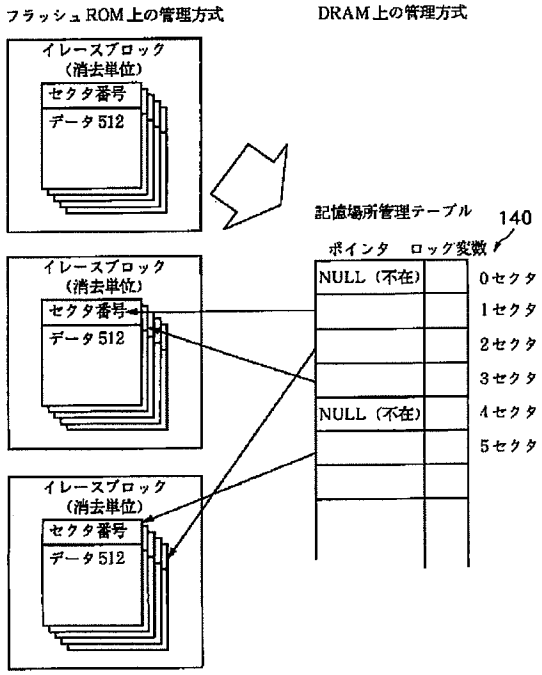
【図8】


(A)
データの移動
(B)
イレース
ブロックの消去
(C)
イレースブロック(1)
消去回数＝5
8 (使用済)
5 (使用中)
7 (使用済)
1f (使用中)
8 (使用中)
22 (使用済)
イレースブロック(2)
消去回数＝9
25 (使用済)
9 (使用中)
(未使用)
(未使用)
(未使用)
イレースブロック(1)
消去回数＝5
8 (使用済)
5 (使用済)
7 (使用済)
1f (使用済)
8 (使用済)
22 (使用済)
イレースブロック(2)
消去回数＝9
25 (使用済)
9 (使用中)
5 (使用中)
1f (使用中)
8 (使用中)
(未使用)
イレースブロック(1)
消去回数＝6
(未使用)
(未使用)
(未使用)
(未使用)
(未使用)
イレースブロック(2)
消去回数＝9
25 (使用済)
9 (使用中)
5 (使用中)
1f (使用中)
8 (使用中)
(未使用)

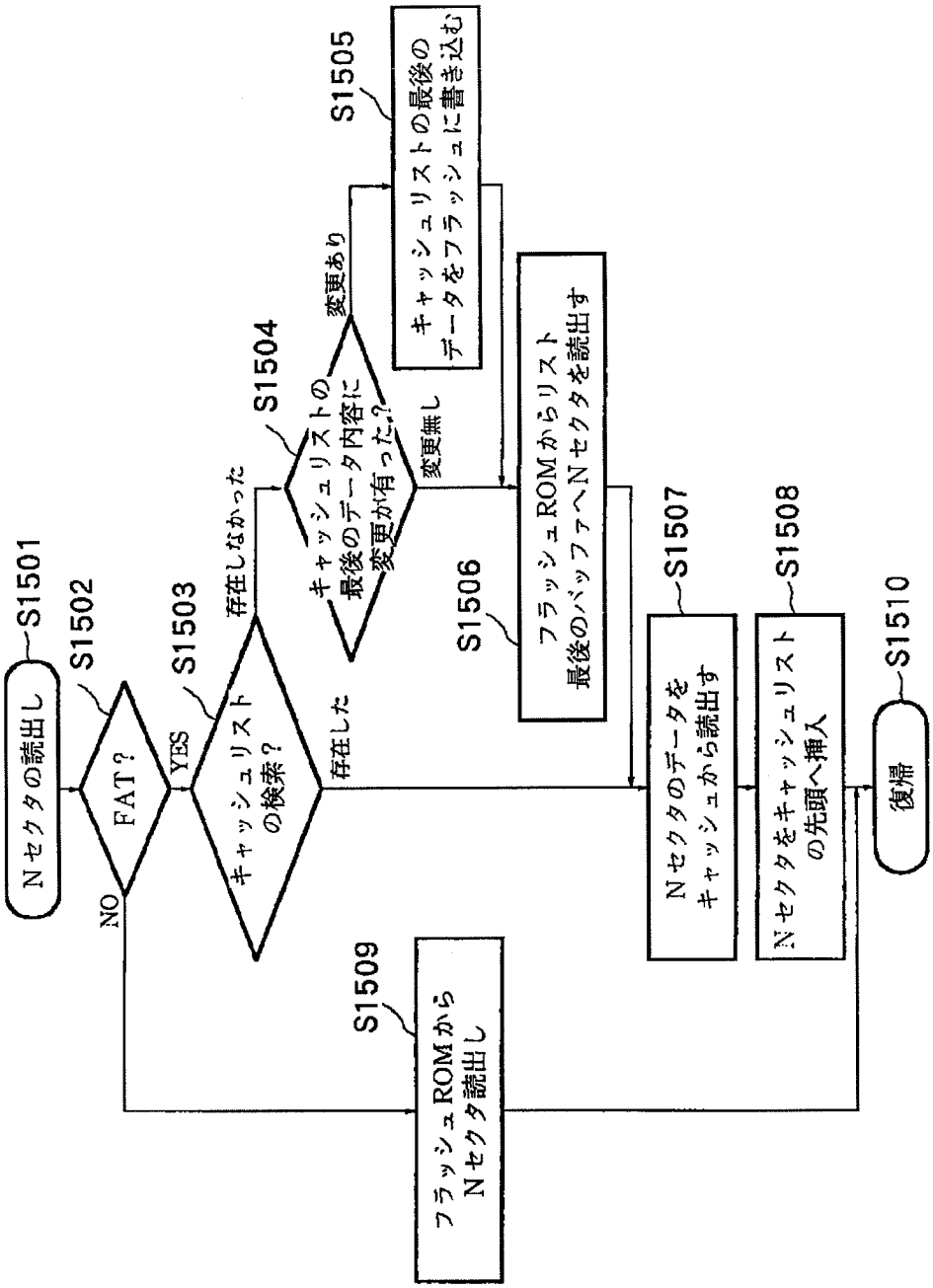
【図9】


イレースブロックの消去
(A)
イレース
ブロック (1)
8 (使用済)
5 (使用中)
7 (使用済)
1f (使用中)
8 (使用中)
22 (使用済)
(C)
イレース
ブロック (1)
(未使用)
(未使用)
(未使用)
(未使用)
(未使用)
(未使用)
(D)
イレース
ブロック (1)
5 (使用中)
1f (使用中)
8 (使用中)
(未使用)
(未使用)
(未使用)
(B)
DRAM
5
1f
8
データの待避
データの復元
14

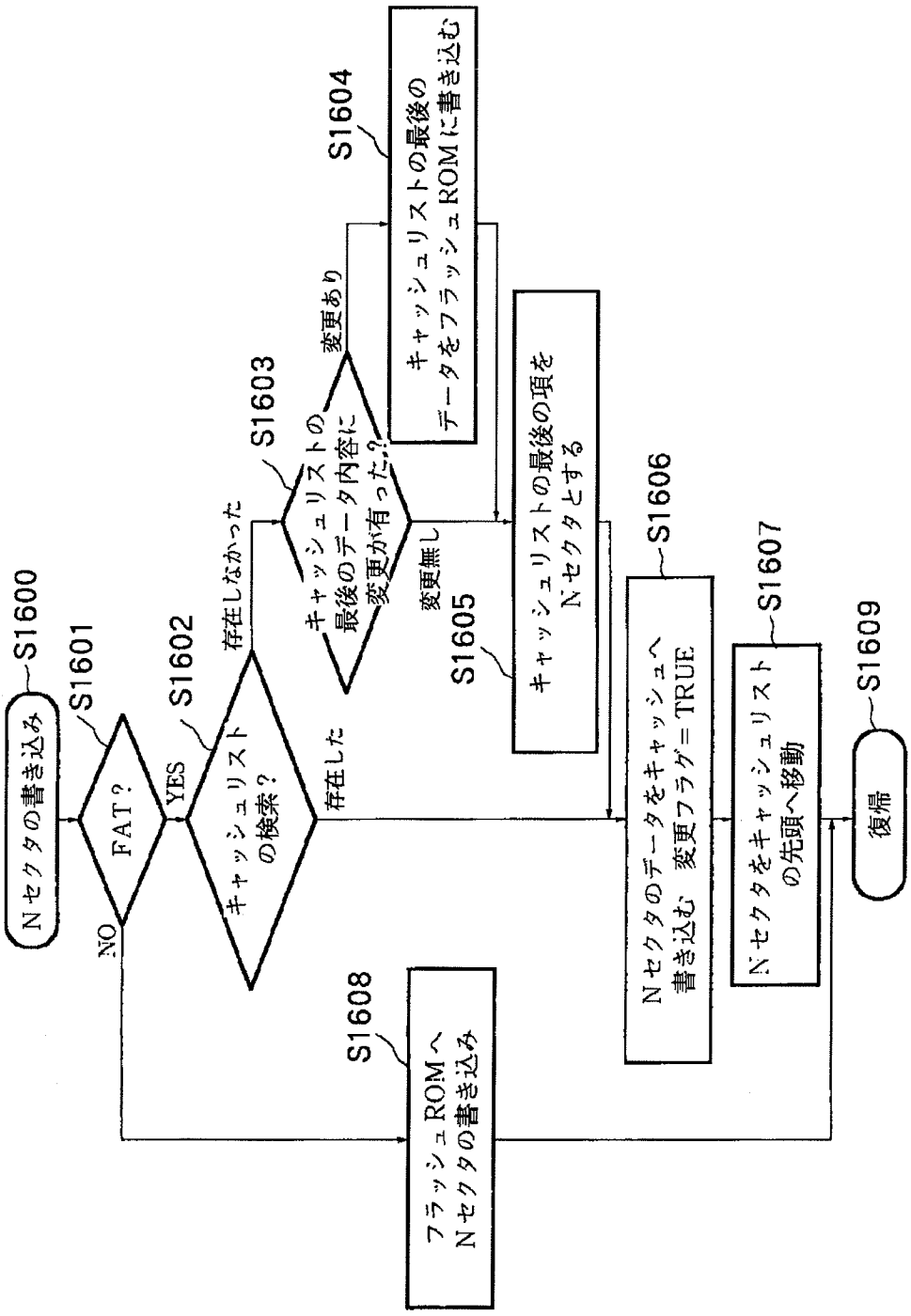
【図10】


フラッシュROM上の管理方式
DRAM上の管理方式
イレースブロック
(消去単位)
セクタ番号
データ512
イレースブロック
(消去単位)
セクタ番号
データ512
イレースブロック
(消去単位)
セクタ番号
データ512
記憶場所管理テーブル
140
ポインタ
ロック変数
NULL (不在)
NULL (不在)
0セクタ
1セクタ
2セクタ
3セクタ
4セクタ
5セクタ

【図13】


Nセクタの読出し
S1501
FAT?
S1502
NO
YES
S1503
キャッシュリストの検索?
存在しなかった
存在した
S1504
キャッシュリストの最後のデータ内容に変更が有った?
変更あり
変更無し
S1505
キャッシュリストの最後のデータをフラッシュに書き込む
S1506
フラッシュROMからリスト最後のバッファへNセクタを読出す
S1507
Nセクタのデータをキャッシュから読出す
S1508
Nセクタをキャッシュリストの先頭へ挿入
S1509
フラッシュROMからNセクタ読出し
S1510
復帰

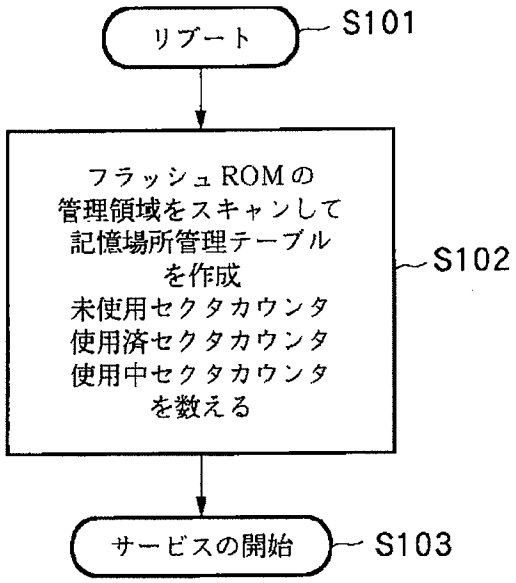
【図１４】


Ｎセクタの書き込み
S1600
FAT？
S1601
NO
YES
キャッシュリストの検索？
S1602
存在しなかった
存在した
キャッシュリストの最後のデータ内容に変更が有った？
S1603
変更あり
変更無し
キャッシュリストの最後のデータをフラッシュROMに書き込む
S1604
キャッシュリストの最後の項をＮセクタとする
S1605
Ｎセクタのデータをキャッシュへ書き込む　変更フラグ＝TRUE
S1606
Ｎセクタをキャッシュリストの先頭へ移動
S1607
フラッシュROMへＮセクタの書き込み
S1608
復帰
S1609

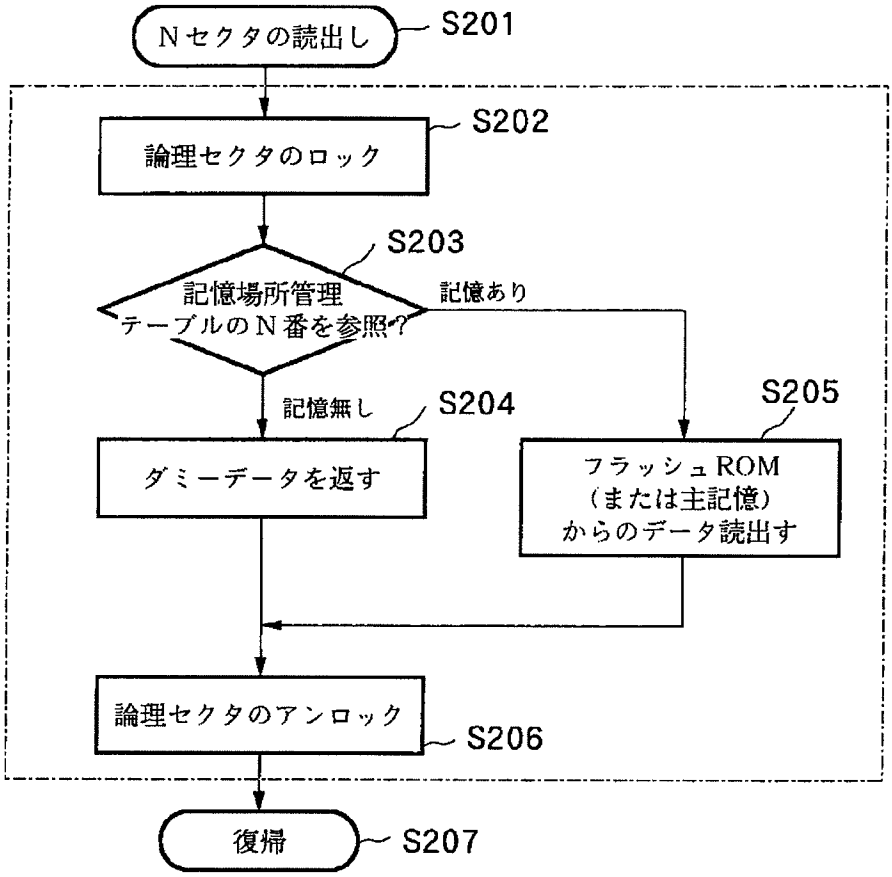
【図16】

```
1 : void ZoomUp (void)
2 : {
3 :     WaitSemapho (SemDCDC) :
4 :     Motor1StepUp (void) :
5 :     SignalSemapho (SemDCDC) :
6 : }
7 : char XWriteSectorEEP (struct * Dev,int Sector ,char * Buffer)
8 : {
9 :     WaitSemapho (SemDCDC) :
10 :    WriteSectorEEP (Dev, Sector, Buffer)
11 :    While ( * Dev - > Address ! = Dev - > Data) :
12 :    SignalSemapho (SemDCDC) :
13 : }
```

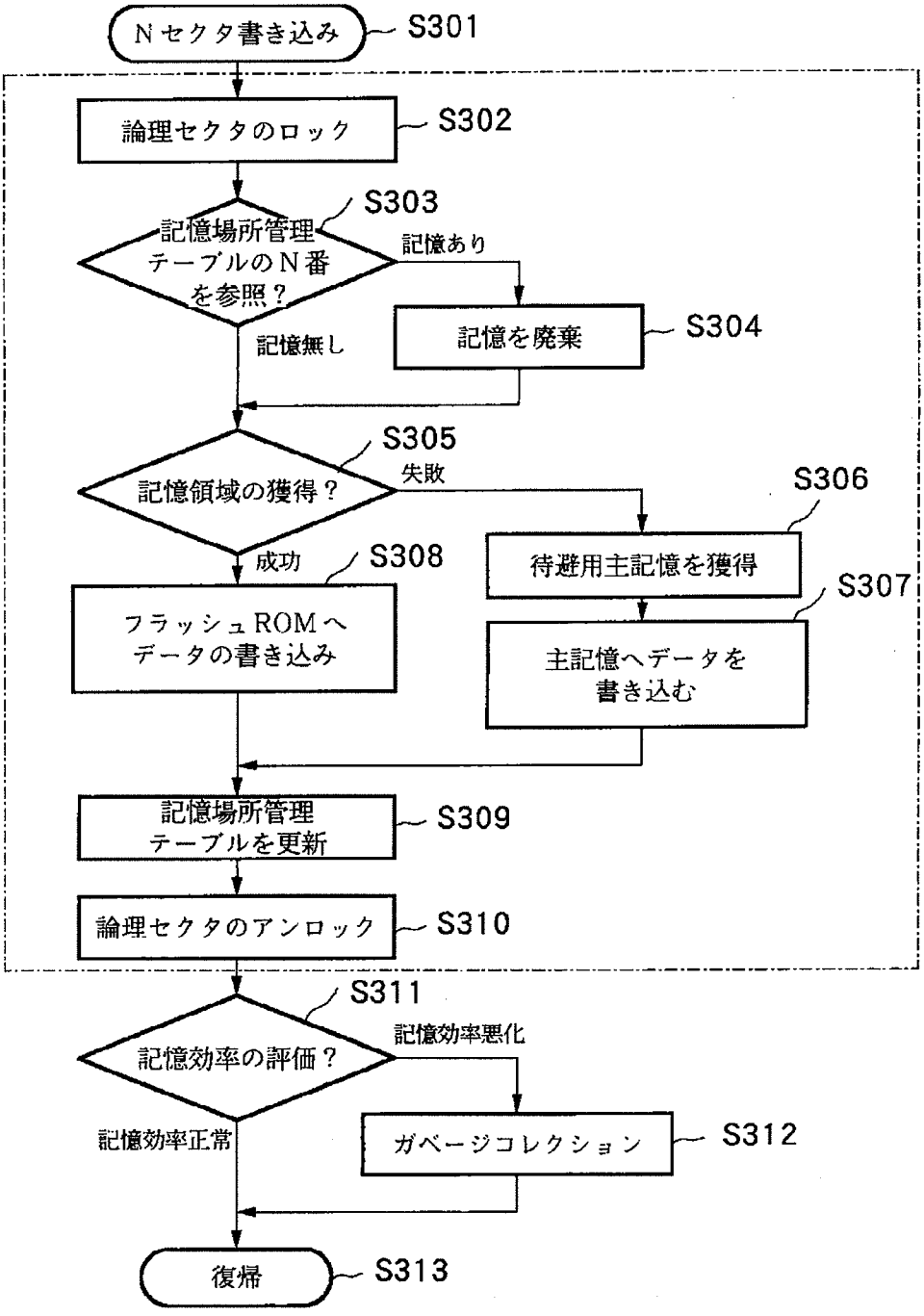
【図17】



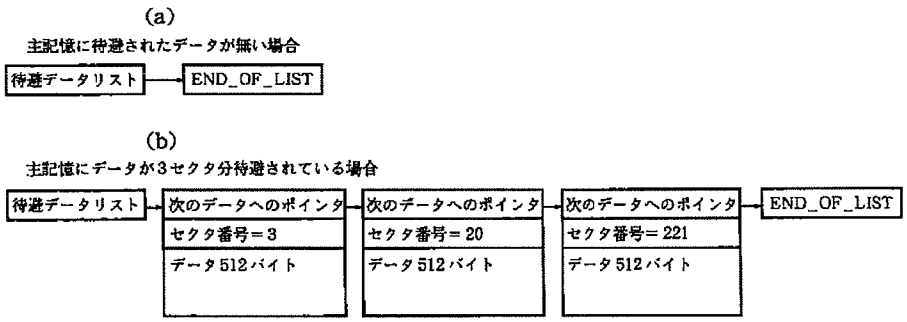
【図18】



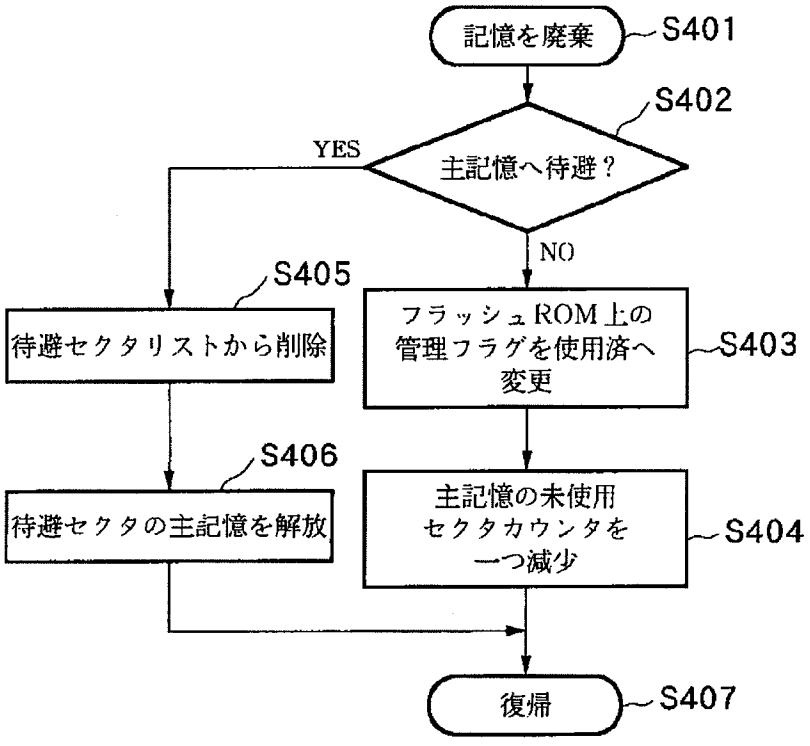
【図１９】


Ｎセクタ書き込み
S301
論理セクタのロック
S302
記憶場所管理テーブルのＮ番を参照？
S303
記憶あり
記憶を廃棄
S304
記憶無し
記憶領域の獲得？
S305
失敗
S306
待避用主記憶を獲得
S307
主記憶へデータを書き込む
成功
S308
フラッシュROMへデータの書き込み
記憶場所管理テーブルを更新
S309
論理セクタのアンロック
S310
記憶効率の評価？
S311
記憶効率悪化
記憶効率正常
ガベージコレクション
S312
復帰
S313

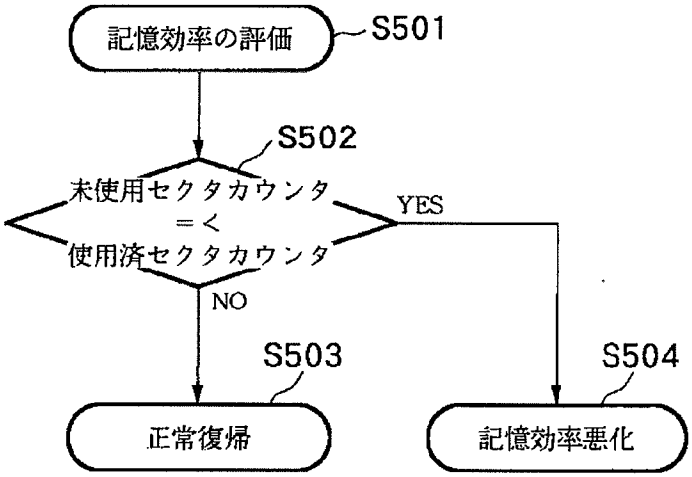
【図２０】


(a)
主記憶に待避されたデータが無い場合
待避データリスト
END_OF_LIST
(b)
主記憶にデータが3セクタ分待避されている場合
待避データリスト
次のデータへのポインタ
セクタ番号＝3
データ512バイト
次のデータへのポインタ
セクタ番号＝20
データ512バイト
次のデータへのポインタ
セクタ番号＝221
データ512バイト
END_OF_LIST

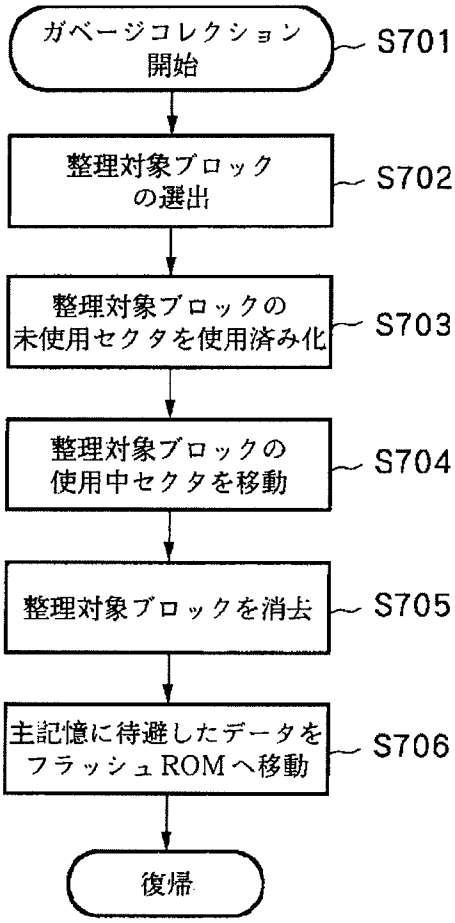
【図２１】


記憶を廃棄
S401
S402
主記憶へ待避？
YES
NO
S405
待避セクタリストから削除
フラッシュROM上の
管理フラグを使用済へ
変更
S403
S406
待避セクタの主記憶を解放
主記憶の未使用
セクタカウンタを
一つ減少
S404
復帰
S407

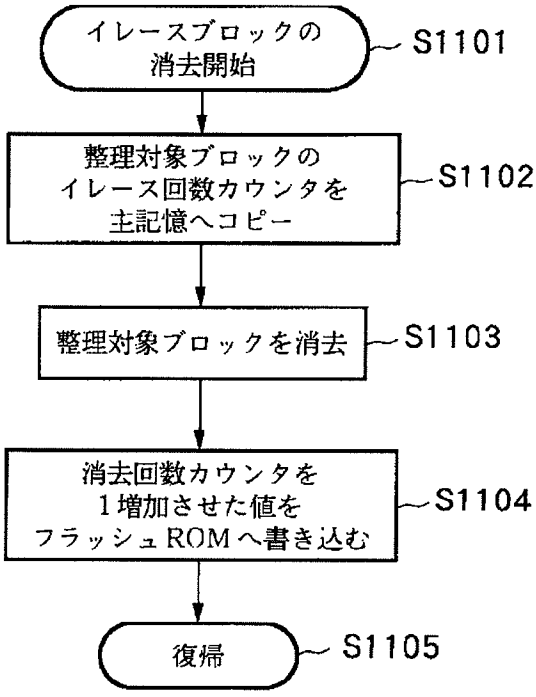
【図２２】



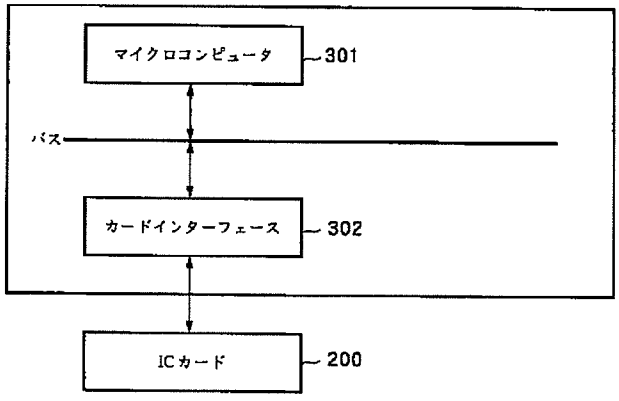
【図２４】



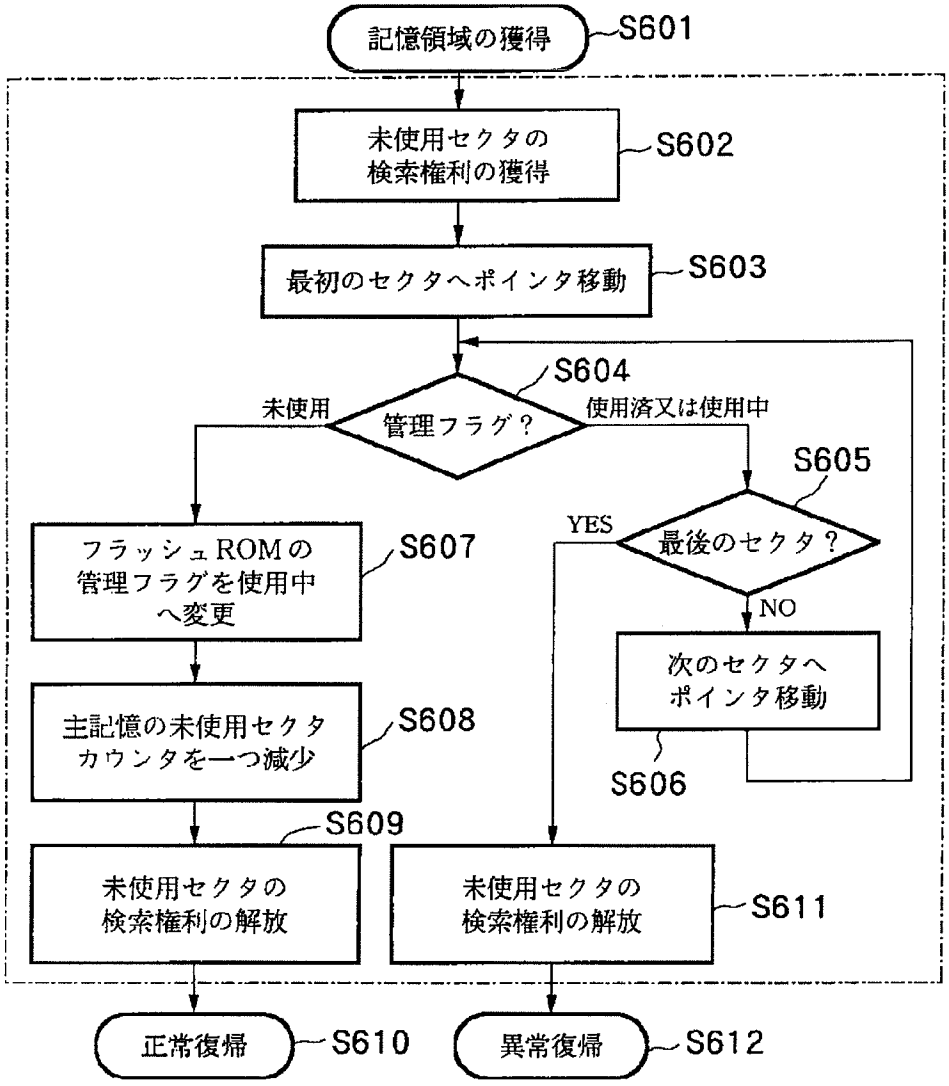
【図２８】



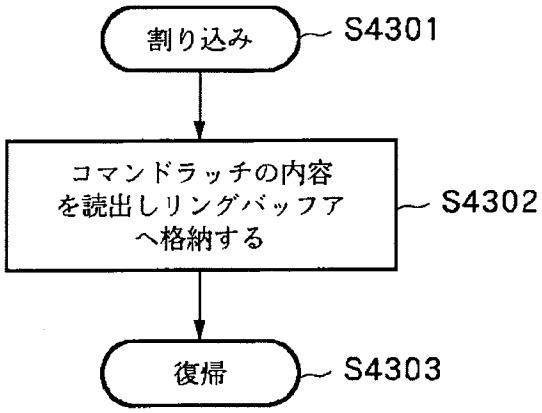
【図３１】



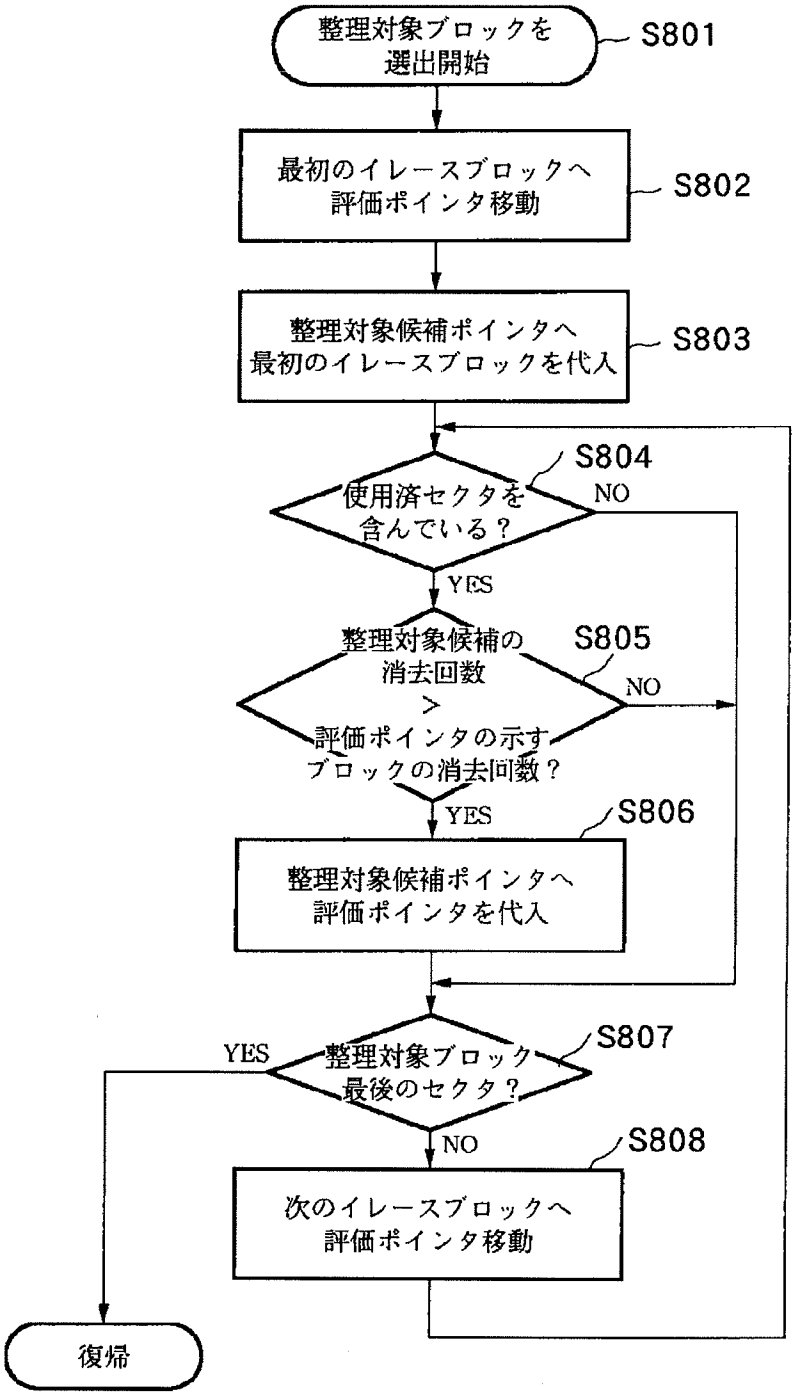
【図２３】


記憶領域の獲得
S601
未使用セクタの
検索権利の獲得
S602
最初のセクタへポインタ移動
S603
S604
管理フラグ？
未使用
使用済又は使用中
S605
最後のセクタ？
YES
NO
フラッシュROMの
管理フラグを使用中
へ変更
S607
次のセクタへ
ポインタ移動
S606
主記憶の未使用セクタ
カウンタを一つ減少
S608
S609
未使用セクタの
検索権利の解放
未使用セクタの
検索権利の解放
S611
正常復帰
S610
異常復帰
S612

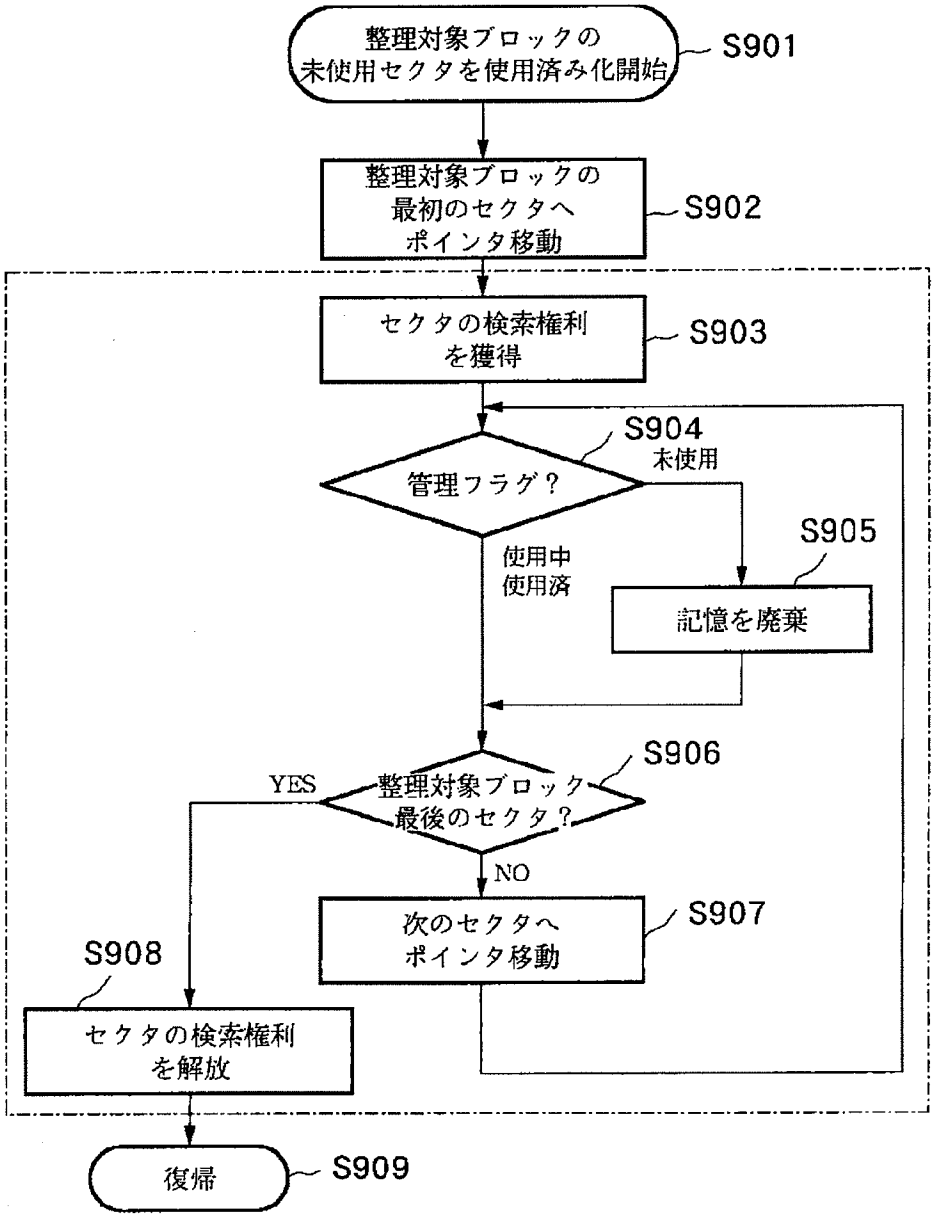
【図３４】


割り込み
S4301
コマンドラッチの内容
を読出しリングバッファ
へ格納する
S4302
復帰
S4303

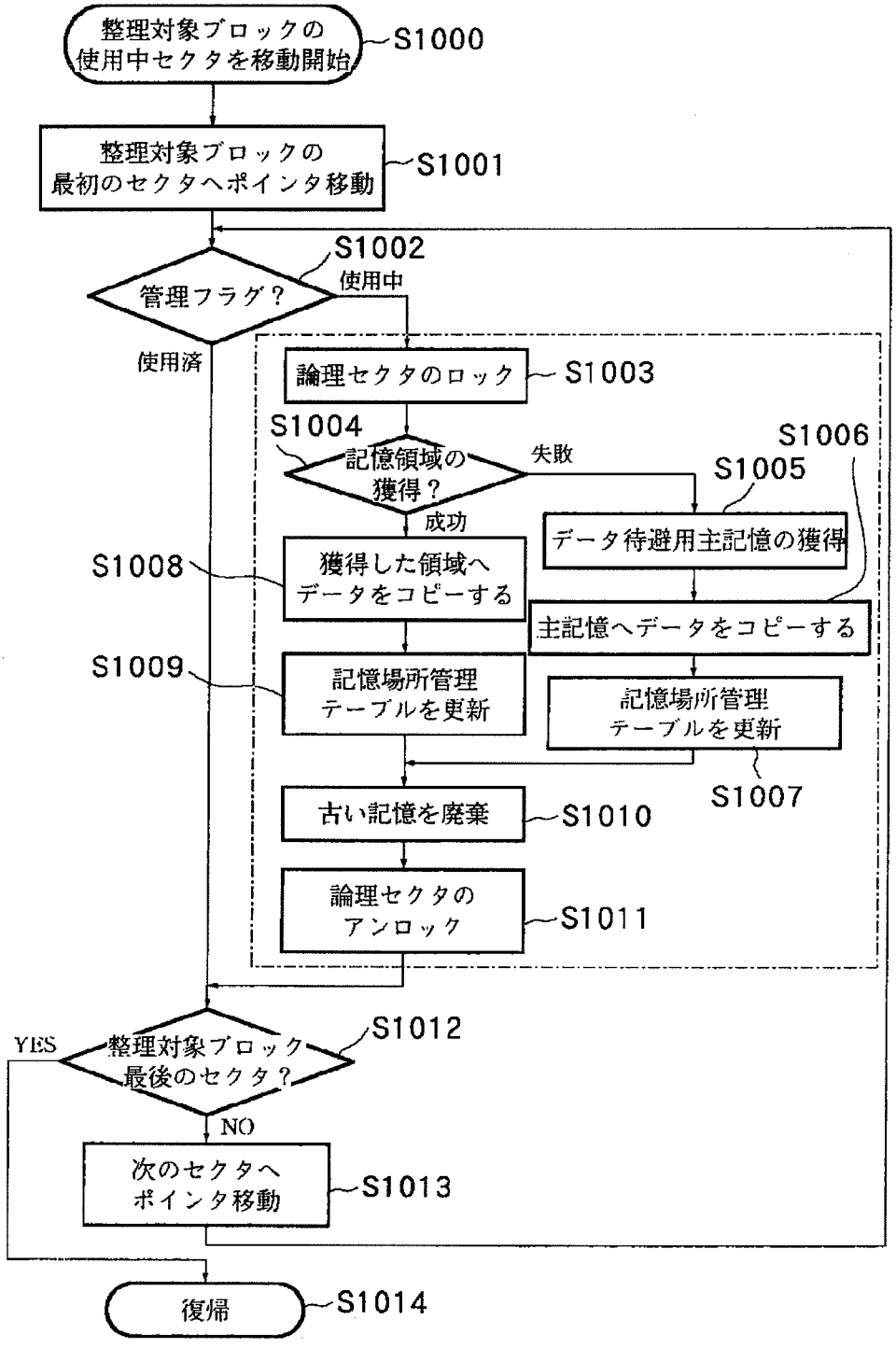
【図２５】


整理対象ブロックを
選出開始
S801
最初のイレースブロックへ
評価ポインタ移動
S802
整理対象候補ポインタへ
最初のイレースブロックを代入
S803
使用済セクタを
含んでいる？
S804
NO
YES
整理対象候補の
消去回数
＞
評価ポインタの示す
ブロックの消去回数？
S805
NO
YES
整理対象候補ポインタへ
評価ポインタを代入
S806
整理対象ブロック
最後のセクタ？
S807
YES
NO
次のイレースブロックへ
評価ポインタ移動
S808
復帰

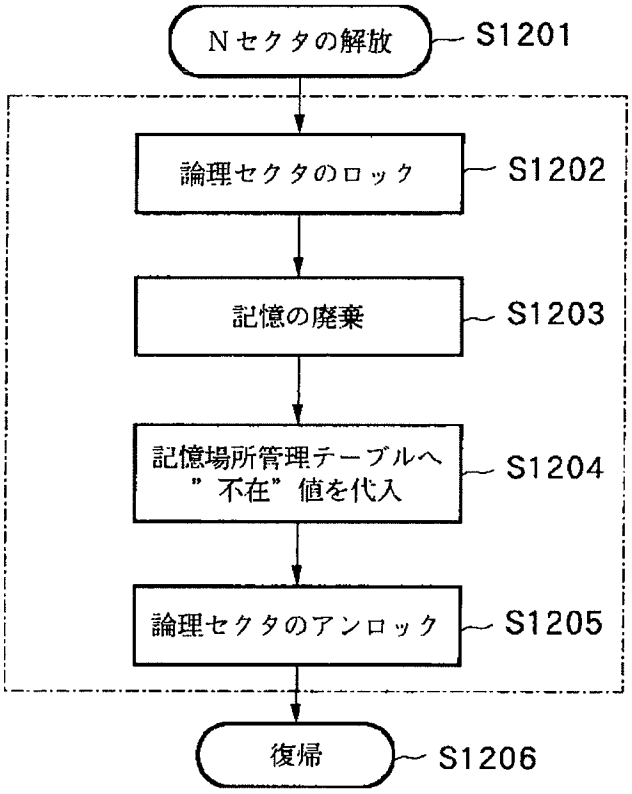
【図26】


整理対象ブロックの
未使用セクタを使用済み化開始
S901
整理対象ブロックの
最初のセクタへ
ポインタ移動
S902
セクタの検索権利
を獲得
S903
S904
管理フラグ？
未使用
使用中
使用済
S905
記憶を廃棄
YES
整理対象ブロック
最後のセクタ？
S906
NO
次のセクタへ
ポインタ移動
S907
S908
セクタの検索権利
を解放
復帰
S909

【図27】


整理対象ブロックの
使用中セクタを移動開始
S1000
整理対象ブロックの
最初のセクタへポインタ移動
S1001
S1002
管理フラグ？
使用中
使用済
論理セクタのロック
S1003
S1004
記憶領域の
獲得？
失敗
S1005
S1006
成功
データ待避用主記憶の獲得
S1008
獲得した領域へ
データをコピーする
主記憶へデータをコピーする
S1009
記憶場所管理
テーブルを更新
記憶場所管理
テーブルを更新
S1007
古い記憶を廃棄
S1010
論理セクタの
アンロック
S1011
YES
整理対象ブロック
最後のセクタ？
S1012
NO
次のセクタへ
ポインタ移動
S1013
復帰
S1014

【図２９】



【図３５】

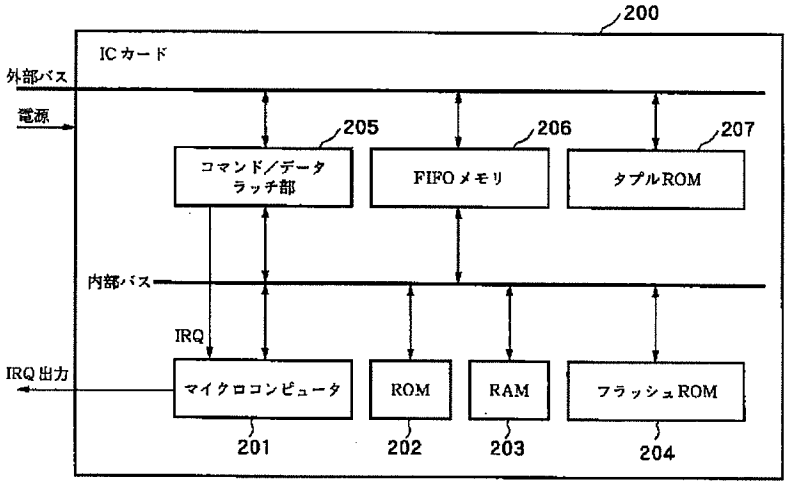
コマンド

読出し (In)	書き込み (Out)	
Error	Features	
SectorCount	SectorCount	
SectorNumber	SectorNumber	
CylinderLow	CylinderLow	
CylinderHigh	CylinderHigh	
Drive／Head	Drive／Head	
Status	Command	←割り込み発生

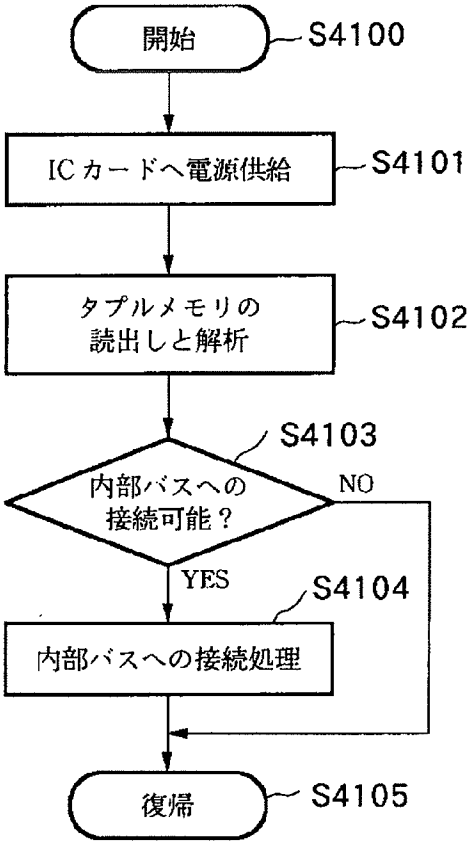
コントロール

読出し (In)	書き込み (Out)
Alt.Status	Device Ctl
DriveAddr	

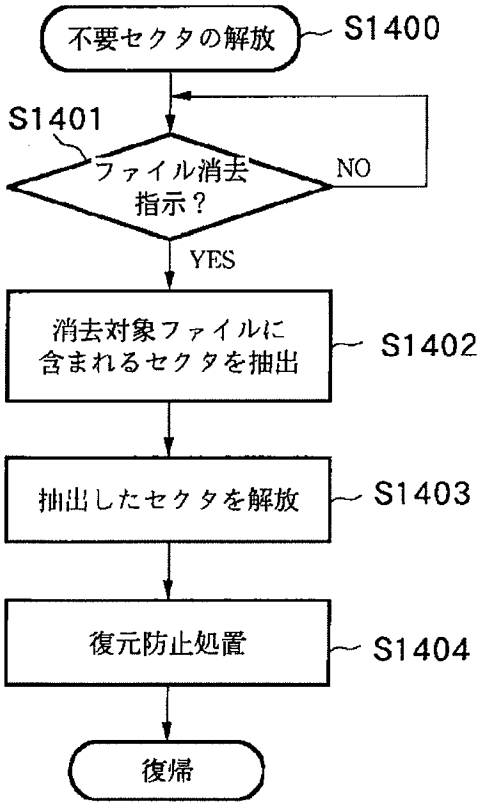
【図３０】



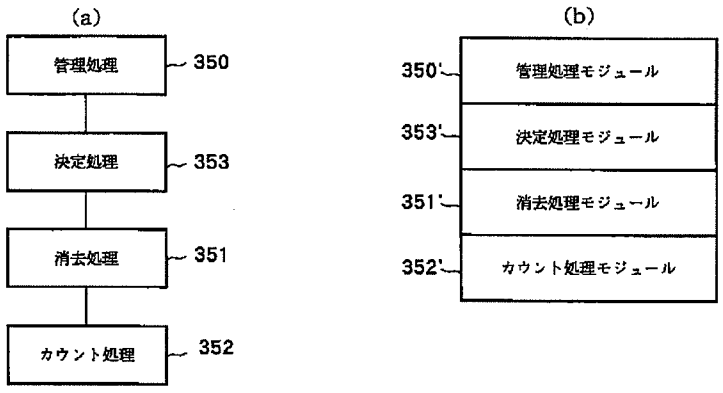
【図３２】


開始
S4100
ICカードへ電源供給
S4101
タプルメモリの
読出しと解析
S4102
S4103
内部バスへの
接続可能？
NO
YES
S4104
内部バスへの接続処理
復帰
S4105

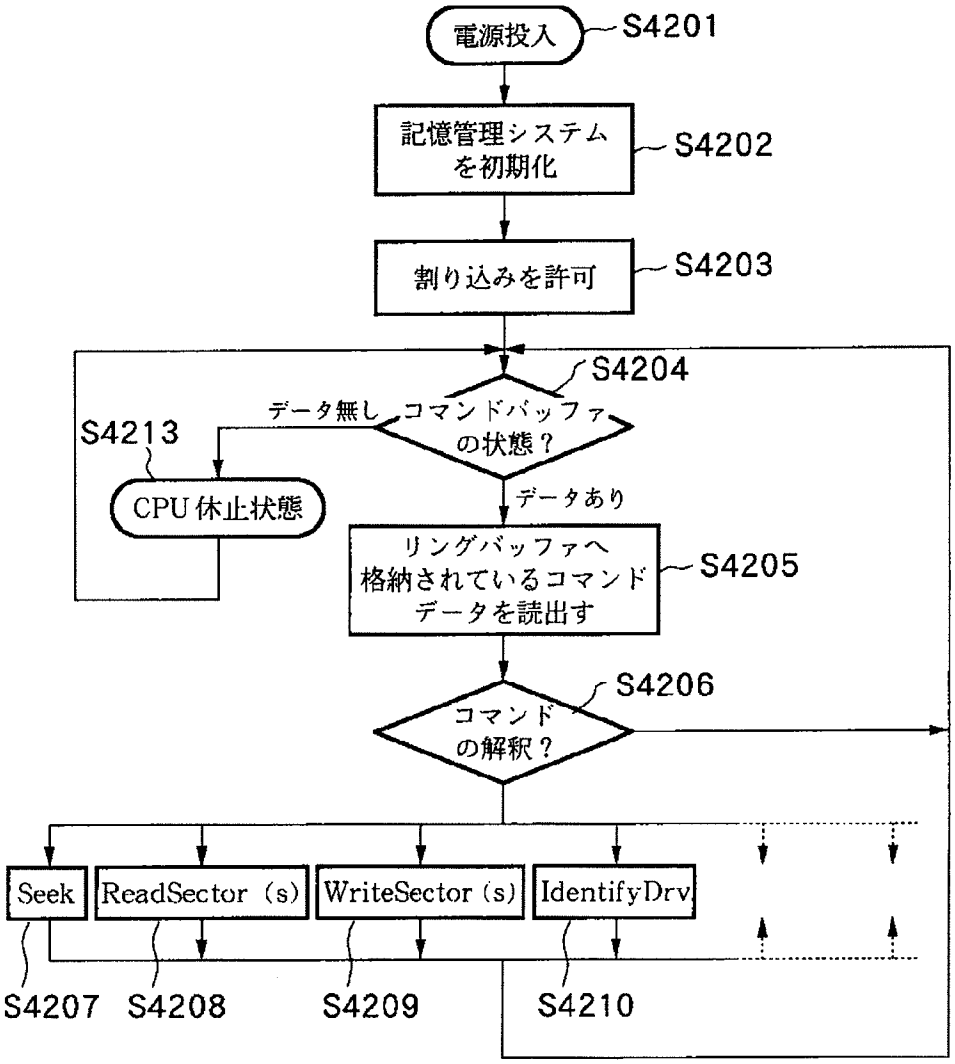
【図３７】


不要セクタの解放
S1400
S1401
ファイル消去
指示？
NO
YES
消去対象ファイルに
含まれるセクタを抽出
S1402
抽出したセクタを解放
S1403
復元防止処置
S1404
復帰

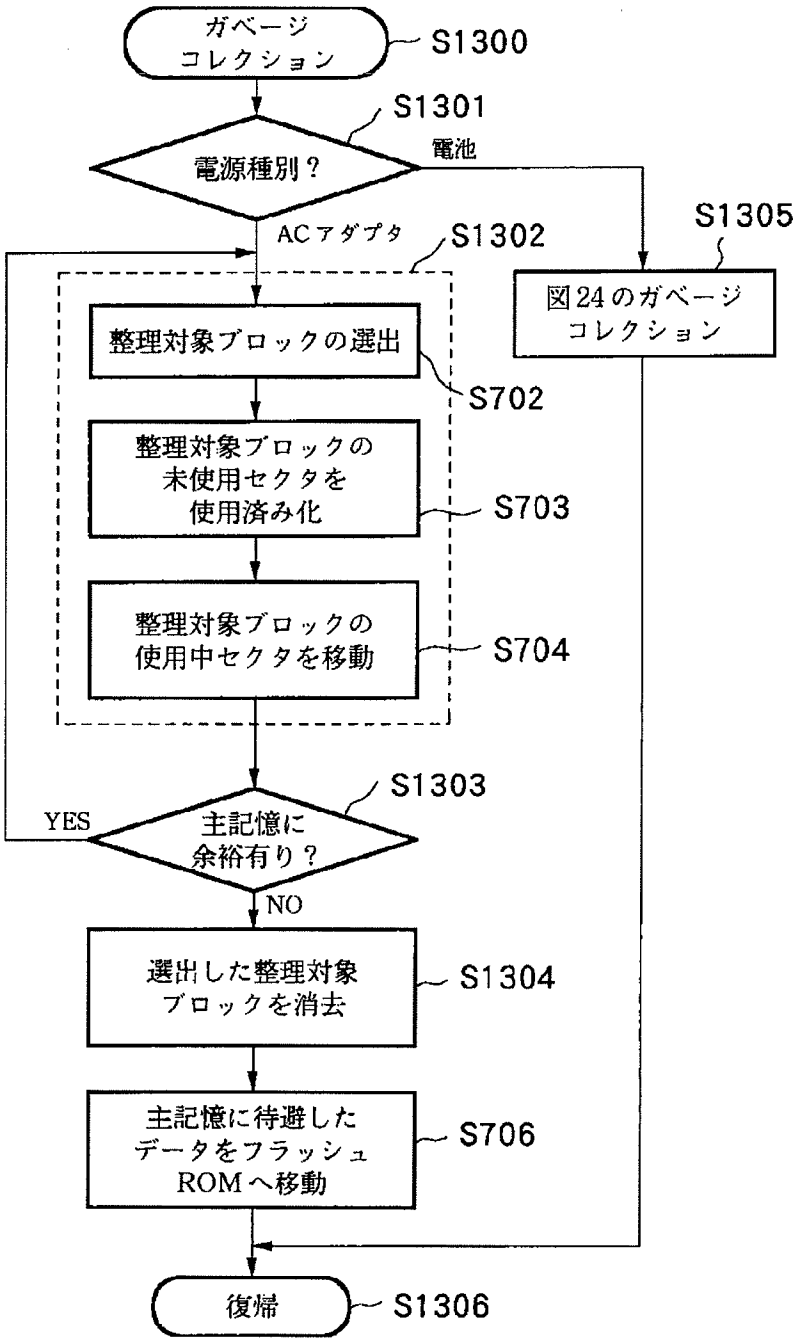
【図４９】


(a)
管理処理
350
決定処理
353
消去処理
351
カウント処理
352
(b)
350'
管理処理モジュール
353'
決定処理モジュール
351'
消去処理モジュール
352'
カウント処理モジュール

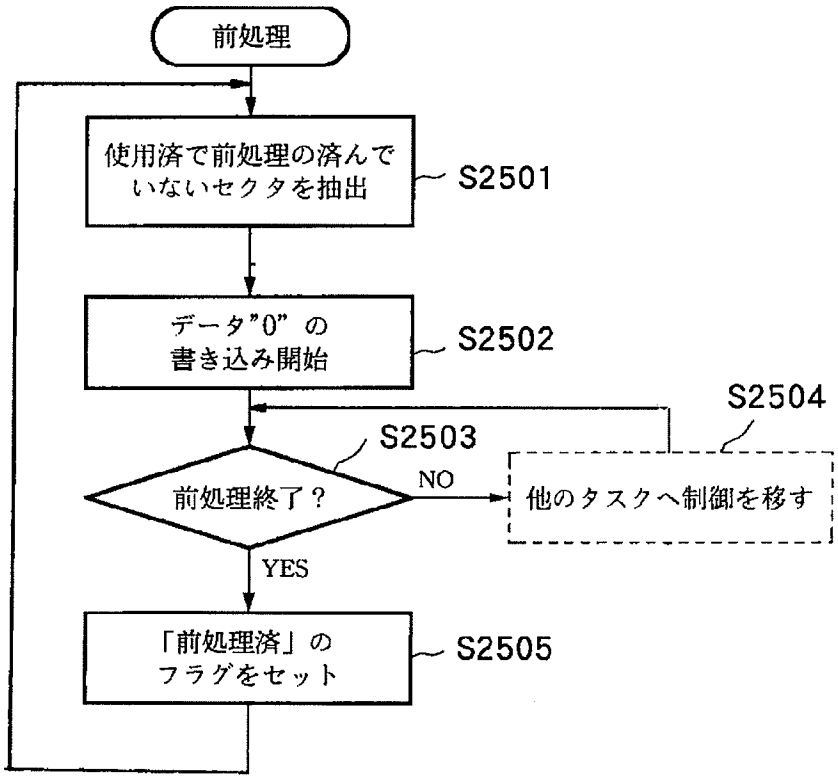
【図３３】


電源投入
S4201
記憶管理システム
を初期化
S4202
割り込みを許可
S4203
S4204
コマンドバッファ
の状態？
データ無し
S4213
CPU 休止状態
データあり
リングバッファへ
格納されているコマンド
データを読出す
S4205
S4206
コマンド
の解釈？
Seek
ReadSector (s)
WriteSector (s)
IdentifyDrv
S4207
S4208
S4209
S4210

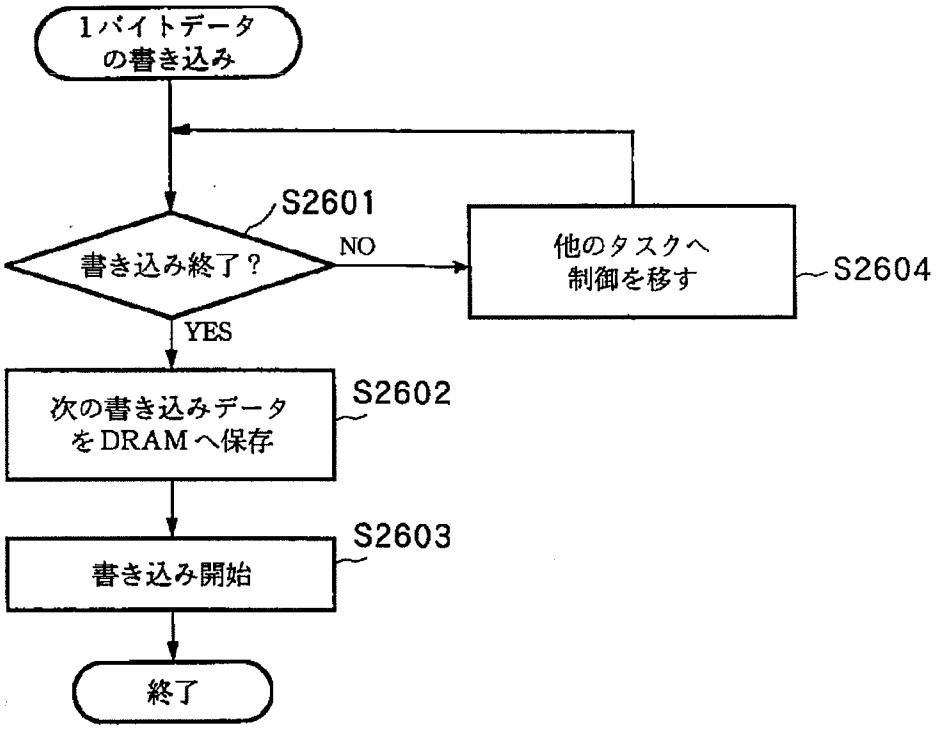
【図36】


ガベージ
コレクション
S1300
S1301
電源種別？
電池
ACアダプタ
S1302
S1305
図24のガベージ
コレクション
整理対象ブロックの選出
S702
整理対象ブロックの
未使用セクタを
使用済み化
S703
整理対象ブロックの
使用中セクタを移動
S704
S1303
YES
主記憶に
余裕有り？
NO
選出した整理対象
ブロックを消去
S1304
主記憶に待避した
データをフラッシュ
ROMへ移動
S706
復帰
S1306

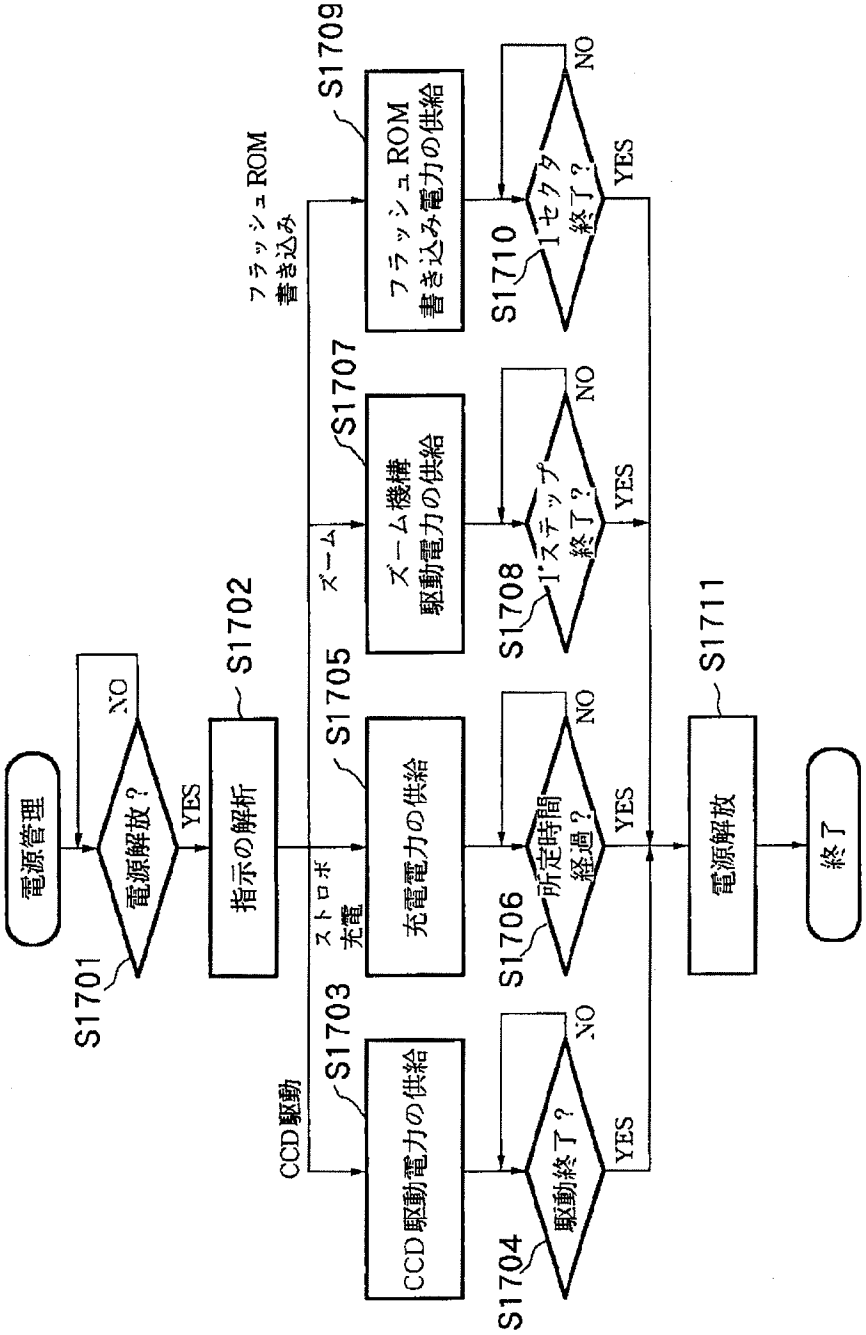
【図38】


前処理
使用済で前処理の済んでいないセクタを抽出
S2501
データ"0"の書き込み開始
S2502
S2504
S2503
前処理終了？
NO
他のタスクへ制御を移す
YES
「前処理済」のフラグをセット
S2505

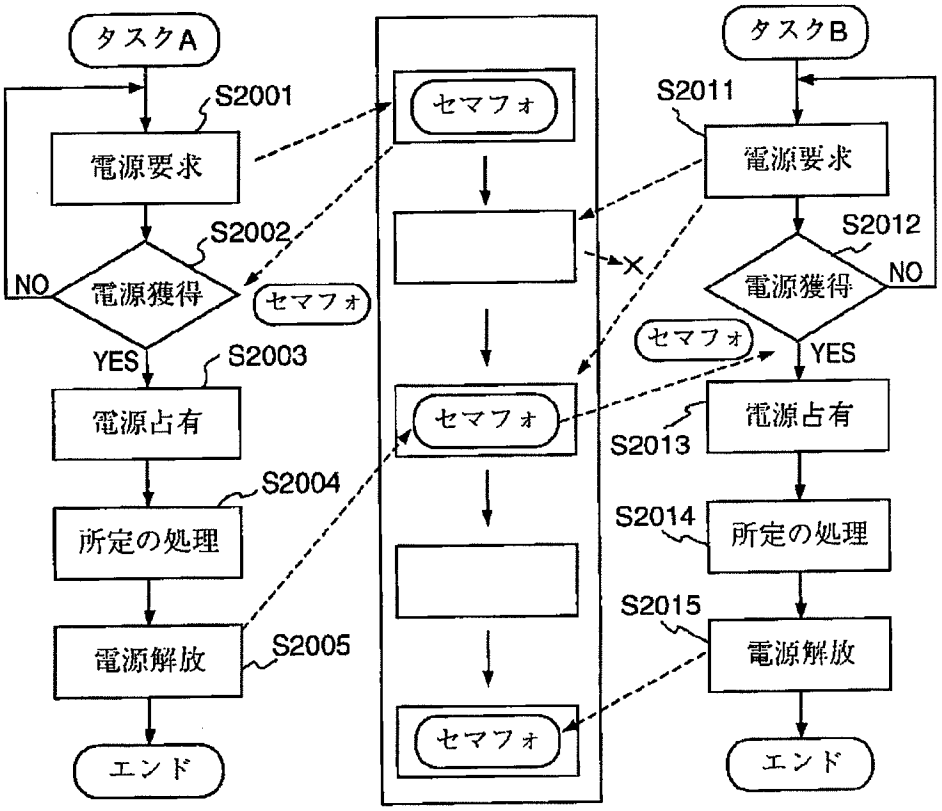
【図39】


1バイトデータ
の書き込み
S2601
書き込み終了？
NO
他のタスクへ
制御を移す
S2604
YES
次の書き込みデータ
をDRAMへ保存
S2602
書き込み開始
S2603
終了

【図40】


電源管理
S1701
電源解放？
NO
YES
S1702
指示の解析
ストロボ
充電
S1705
充電電力の供給
S1706
所定時間
経過？
NO
YES
CCD駆動
S1703
CCD駆動電力の供給
S1704
駆動終了？
NO
YES
ズーム
S1707
ズーム機構
駆動電力の供給
S1708
1ステップ
終了？
NO
YES
フラッシュROM
書き込み
S1709
フラッシュROM
書き込み電力の供給
S1710
1セクタ
終了？
NO
YES
S1711
電源解放
終了

【図48】


タスクA
S2001
電源要求
S2002
NO
電源獲得
セマフォ
YES
S2003
電源占有
S2004
所定の処理
電源解放
S2005
エンド
セマフォ
セマフォ
セマフォ
タスクB
S2011
電源要求
S2012
電源獲得
NO
セマフォ
YES
電源占有
S2013
S2014
所定の処理
S2015
電源解放
エンド